

Mechanical Control of a Wheelchair by means of EEG Signals

by
Ian Nicolaas van Wyk

*Thesis presented in partial fulfilment of the requirements for the degree
of Master of Engineering (Mechatronic) in the Faculty of Engineering at
Stellenbosch University*



UNIVERSITEIT
iYUNIVESITHI
STELLENBOSCH
UNIVERSITY

100
1918 · 2018

Supervisor: Dr Dawie van den Heever

March 2018

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: March 2018

Copyright © 2018 Stellenbosch University
All rights reserved



UNIVERSITEIT • STELLENBOSCH • UNIVERSITY
jou kennisvennoot • your knowledge partner

Plagiaatverklaring / Plagiarism Declaration

- 1 Plagiaat is die oorneem en gebruik van die idees, materiaal en ander intellektuele eiendom van ander persone asof dit jou eie werk is.
Plagiarism is the use of ideas, material and other intellectual property of another's work and to present it as my own.
- 2 Ek erken dat die pleeg van plagiaat 'n strafbare oortreding is aangesien dit 'n vorm van diefstal is.
I agree that plagiarism is a punishable offence because it constitutes theft.
- 3 Ek verstaan ook dat direkte vertalings plagiaat is.
I also understand that direct translations are plagiarism.
- 4 Dienooreenkomstig is alle aanhalings en bydraes vanuit enige bron (ingesluit die internet) volledig verwys (erken). Ek erken dat die woordelike aanhaal van teks sonder aanhalingstekens (selfs al word die bron volledig erken) plagiaat is.
Accordingly all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism.
- 5 Ek verklaar dat die werk in hierdie skryfstuk vervat, behalwe waar anders aangedui, my eie oorspronklike werk is en dat ek dit nie vantevore in die geheel of gedeeltelik ingehandig het vir bepunting in hierdie module/werkstuk of 'n ander module/werkstuk nie.
I declare that the work contained in this assignment, except where otherwise stated, is my original work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/assignment.

| | |
|--|---------------------------------|
| Studentenommer / Student number | Handtekening / Signature |
| Voorletters en van / Initials and surname | Datum / Date |

Abstract

The aim of this project was to incorporate a low cost electric wheelchair by modifying an existing self-propelled one with a neuro headset. The headset must control the wheelchair by using EEG data and motor imagery. Binary and discriminant analysis classifiers were implemented with accuracies ranging between 44% and 68.75%. It is concluded that dynamic classification has a very low accuracy compared to using pre acquired data. Also motor imagery is not very well suited when used with the Epoc+ neuro headset. The wheelchair was never constructed, however it was concluded that the listed components are sufficient to create a cheaper alternative for an electric wheelchair.

Abstrak

Die doel van die projek was om 'n lae koste elektriese rolstoel te inkorporeer deur 'n bestaande stoot-rolstoel te modifiseer. Hierdie rolstoel moet deur 'n neuro-kopstuk beheer word deur middel van EEG seine en motor beelde. Binêre en diskriminante analise klassifikasies was gebruik met akkuraathede tussen 44% en 68.75%. Hieruit word afgelei dat dinamiese klassifikasie minder akkuraat is as wanneer vooraf bepaalde waardes gebruik word. Ook is motor beelde nie 'n baie goeie keuse wanneer die Epoc+ neuro-kopstuk gebruik word nie. Die rolstoel was nooit gebou nie, maar daar is afgelei dat die componente gelys in die tesis voldoende is om 'n goedkoop alternatief vir 'n elektriese rolstoel te bou.

Acknowledgements

The author would like to thank the following people for their contribution

- Dawie van den Heever, for being an enthusiastic mentor and leader
- Wayne Swart for his knowledge about mechanical components
- My parents, Arras and Marian van Wyk for their continued love and support

Table of Contents

| | |
|---|-----|
| Declaration..... | i |
| Abstract..... | iii |
| Abstrak..... | iv |
| Acknowledgements..... | v |
| Table of Contents..... | vi |
| List of Figures..... | ix |
| List of Tables | xi |
| Nomenclature..... | xii |
| 1. Introduction..... | 1 |
| 1.1 Problem Statement | 1 |
| 1.2 Solution | 1 |
| 1.3 Previous Work..... | 2 |
| 1.4 Project Overview..... | 2 |
| 2. Literature Studies | 3 |
| 2.1 Electroencephalography | 3 |
| 2.1.1 EEG Signals | 3 |
| 2.1.2 10-20 Node Placement..... | 7 |
| 2.1.3 Epoc+ | 9 |
| 2.2 Brain-Computer Interface | 12 |
| 2.2.1 P300 | 12 |
| 2.2.2 Visually Evoked Potentials | 13 |
| 2.2.3 Motor Imagery | 14 |
| 2.3 DC Motors..... | 15 |
| 2.3.1 Brushless DC Motors vs Brushed DC Motors..... | 15 |
| 2.3.2 Motor Controller | 16 |
| 2.4 System Controller..... | 17 |
| 2.5 Pattern Recognition Techniques | 18 |
| 2.5.1 Linear Discriminant Analysis | 18 |
| 2.5.2 Binary Threshold Classification | 20 |

| | | |
|-------|-----------------------------------|----|
| 2.7 | Available Software..... | 21 |
| 3. | Wheelchair | 23 |
| 3.1 | Engineering Specifications..... | 23 |
| 3.2 | Design | 23 |
| 3.2.1 | Wheels | 24 |
| 3.2.2 | Motors..... | 25 |
| 3.2.3 | Gearbox..... | 28 |
| 3.2.4 | Battery..... | 31 |
| 3.2.5 | Additional Designs | 32 |
| 3.3 | Wheelchair Assembly | 34 |
| 4. | Controller Hardware | 35 |
| 4.1 | Raspberry Pi | 36 |
| 4.2 | Electronic Speed Controllers..... | 37 |
| 4.3 | Central Processing Unit..... | 39 |
| 4.4 | Epoc+ | 39 |
| 4.5 | System Connections | 39 |
| 5. | Controller Software..... | 41 |
| 5.1 | Serial Communication..... | 41 |
| 5.2 | Raspberry Pi Code..... | 42 |
| 5.3 | Acquired Data | 45 |
| 5.3.1 | Training Data | 45 |
| 5.3.2 | Test Data | 45 |
| 5.4 | Signal Processing | 46 |
| 5.5 | Classification Software | 47 |
| 5.5.1 | Binary Classification..... | 47 |
| 5.5.2 | LDA | 48 |
| 5.6 | Program Flow | 49 |
| 6. | Results..... | 50 |
| 6.1 | Battery Life | 50 |
| 6.2 | Classification Algorithm | 50 |
| 6.2.1 | Timing..... | 50 |

| | |
|---|----|
| 6.2.2 Accuracy | 51 |
| 6.3 Wheelchair Response | 53 |
| 7. Conclusions and Recommendations | 56 |
| 8. References..... | 58 |
| Appendix A: User Manual for the Epoc+ | 63 |
| Appendix B: Motor Calculations..... | 63 |
| Appendix C Source Code Snippets | 65 |
| Appendix C1: Linear Discriminant Analysis | 65 |
| Appendix C2: Binary Threshold | 65 |
| Appendix C3: Raspberry Pi Code | 66 |
| Appendix D Engineering Drawings..... | 73 |
| Appendix E List of Programs | 76 |
| Appendix F EEG States | 76 |

List of Figures

| | |
|---|----|
| Figure 1: A Typical Motor Neuron (Marieb, 2015)..... | 3 |
| Figure 2: Nerve Impulse propagation (Marieb, 2015) | 5 |
| Figure 3: Membrane Potential vs Time (Transmission of Nerve Impulses, 2017).. | 5 |
| Figure 4: Impacts of Synchronization and desynchronization of Neurons on EEG Signals (Kirschstein and Köhling, 2009) | 6 |
| Figure 5: Cerebral Cortex (Sterling, 2017) | 8 |
| Figure 6: 10-20 Node Placement System (Oostenveld and Praamstra, 2001)..... | 8 |
| Figure 7: EEG Bands Intensity on the Cerebral Cortex (What is a qEEG?, 2017) . | 9 |
| Figure 8: Epoc+ Neuroheadset (Emotiv, 2014) | 10 |
| Figure 9: Electrode Placement (Emotiv, 2014) | 11 |
| Figure 10: P-300 Stimuli (‘Mind-Control’ gaming devices leak users’ secrets, 2012) | 13 |
| Figure 11: Visually Evoked Potential (Aysha and Nadia, 2016)..... | 14 |
| Figure 12: Motor Imagery Delay (Cho et al, 2017)..... | 14 |
| Figure 13: ESC Schematic | 16 |
| Figure 14: Minimum and Maximum Pulse Width of the PWM Signal..... | 17 |
| Figure 15: Pure EEG..... | 21 |
| Figure 16: Emokey..... | 22 |
| Figure 17: Emo Composer..... | 22 |
| Figure 18: Guitel Black, Red Castor Wheel (Guitel Black, Red Castor Wheel, 2017) | 24 |
| Figure 19: LDPower Brushless Outrunner 2820-920KV (LDPower Brushless Outrunner 2820-902Kv) | 26 |
| Figure 20: Relationship between Torque, Speed and Power (Motor Torque Specs Minefield) | 27 |
| Figure 21: Varvel Worm Gearbox (Worm gearbox / right-angle, 2017)..... | 30 |
| Figure 22: Victron Energy 12V 60-Ah-55Ah (Critical Power, 2009)..... | 31 |
| Figure 23: Bar Design..... | 32 |
| Figure 24: Bracket Design | 32 |

| | |
|--|----|
| Figure 25: Key Design | 33 |
| Figure 26: Coupling Top View | 33 |
| Figure 27: Rod Design | 34 |
| Figure 28: Assembly Design..... | 35 |
| Figure 29: Raspberry Pi 3B (Raspberry Pi 3 Model B, 2017) | 36 |
| Figure 30: Hobby King SS Series 50-60A ESC (Hobbyking™ SS Series 50-60A ESC (Opto only), 2016) | 37 |
| Figure 31: System Diagram | 40 |
| Figure 32: Server Ethernet Setup | 41 |
| Figure 33: Client Ethernet Setup | 42 |
| Figure 34: Raspberry Pi 3b Pin Layout (. Pin Numbering - Raspberry Pi 3 Model B, 2016) | 42 |
| Figure 35: Maximum Pulse Width Output (Left) and Minimum Pulse Width Output (Right) at Pin 12 of the Raspberry Pi..... | 44 |
| Figure 36: Raspberry Pi Code in Case of Forward Movement..... | 44 |
| Figure 37: Code Snippet for Subtracting Means | 46 |
| Figure 38: EEG Signals of Idle State..... | 46 |
| Figure 39: Program Flow for Raspberry Pi and Laptop | 49 |
| Figure 40: Test Setup of Entire System | 54 |
| Figure 41: PWM Outputs on Oscilloscope | 55 |
| Figure 42: Torque vs Speed | 64 |
| Figure 43: Bar Drawing | 73 |
| Figure 44: Bracket Drawing | 74 |
| Figure 45: Key Drawing | 74 |
| Figure 46: Rod Drawing | 75 |
| Figure 47: Assembly Drawing..... | 75 |
| Figure 48: EEG Signals of Idle State..... | 77 |
| Figure 49: EEG Signals of Forward State | 77 |
| Figure 50: EEG Signals of Stop State..... | 78 |
| Figure 51: EEG Signals of Right State | 78 |
| Figure 52: EEG Signals of Left State | 79 |

List of Tables

| | |
|---|----|
| Table 1: Frequency Bands' Appearances and Mental States (Garcia et al, 2014) ... | 7 |
| Table 2 : Epoc+ Specifications (Emotiv, 2014)..... | 11 |
| Table 3: Advantages and Disadvantages of Brushless and Brushed DC Motors (Miller, 2010)..... | 15 |
| Table 4: Raspberry Pi 3 Ports | 18 |
| Table 5: Engineering Specifications | 23 |
| Table 6: Wheel specification [Guitel Black, Red Castor Wheel, 2017] | 25 |
| Table 7: Motor specification (LDPower Brushless Outrunner 2820-902Kv) | 26 |
| Table 8: Static Friction Coefficients (Friction and Friction Coefficients, 2017)... | 29 |
| Table 9: Frictional Force, Torque and Gear ratio values for different masses | 29 |
| Table 10: ESC Program Options (User instruction for RC aircraft ESC, 2017) ... | 38 |
| Table 11: Raspberry Pi Commands | 43 |
| Table 12: Binary Values of Training Data..... | 47 |
| Table 13: Sequence of Classification Algorithm testing | 51 |
| Table 14: Confusion Matrix for LDA | 52 |
| Table 15: Confusion Matrix for Binary Classifier | 52 |
| Table 16: Parts and Descriptions of Tested System | 54 |

Nomenclature

| Symbol | Meaning |
|----------------------|--|
| $(X)^{-1}$ | Inverse of a Matrix X |
| $(X)^T$ | Transponent of Matrix X |
| .csv | Comma-separated Value file |
| $\Sigma_{i=0}^k X_i$ | Sum of X components ranging from 0 to k |
| P(X) | Probability of X |
| P(X K) | Conditional Probability of X given K |
| P_{max} | Maximum power of the motor |
| T_{stall} | Maximum torque of the motor while output speed is zero |
| W_{max} | Maximum rotational speed of motor |
| W_{motor} | Current rotational speed of motor |
| W_{wheel} | Current rotational speed of the wheel |

1. Introduction

This project is divided into two different segments. The first part is to construct a low-cost electrical wheelchair by modifying a self-propelled wheelchair. The second part is to use EEG signals to control the newly constructed wheelchair.

1.1 Problem Statement

A quadriplegic is defined as someone who lost control over all 4 limbs of their body. Quadriplegics are extremely dependant on their caregivers and are usually monitored 24 hours a day.

Quadriplegics also have the need to be transported to various locations. Thus there is a need for quadriplegics to be able to move independently to reduce effort required by caregivers and also to give the quadriplegics a sense of independence.

Electric wheelchairs are also very expensive in general which most require a working hand or foot to operate.

1.2 Solution

The solution chosen was to modify a self-propelled wheelchair using low-cost components which can operate by using brain signals. Thorough research must be done on components that can be mounted on the wheelchair to create an efficient yet low-cost electric wheelchair.

The wheelchair must then be able to move forward, left and right by means of the brain signals, more specifically thought evoked signals. The wheelchair will thus be controlled by a neuro-headset. The data generated by the neuroheadset will be stored as certain patterns, and if familiar patterns are recognized can the system command the wheelchair to move.

1.3 Previous Work

Various projects are available where wheelchairs are controlled by using EEG signals, the most popular one being from Instructables (Brain-Controlled Wheelchair, 2016).

There are 2 main differences between the Instructables' project and this one. The first is the difference in wheelchairs. The Instructables used an existing electric wheelchair with a joystick and modified it to be controlled by an Arduino. The second difference is that the Instructables used the programs and detection algorithms created by Emotiv – the company that made the Epoc+ neuroheadset.

1.4 Project Overview

The project starts off with a complete literature study regarding the hardware and software needed for the project as well as studies regarding EEG data. The next step is to design an electrical wheelchair given a manual wheelchair. The chapter starts off identifying feasible components with appropriate motivation and concludes with instructions on how to assemble the power wheelchair given the components.

The next chapter focusses on the hardware required to control the wheelchair. Various items are chosen and choices are motivated. The chapter concludes on how to assemble the controller system. The next step is to incorporate appropriate software that can be run on the controller. The software is responsible for identifying EEG patterns which in turn is used to move the wheelchair. Various signal processing are also done on the EEG signals.

The project concludes with the results given which is compared to the specifications. After the results are satisfactory can relevant conclusions be made and certain recommendations be given.

2. Literature Studies

This section focusses on the research done prior to the design phase of the project. Extensive research was done on both the mechanical and electrical aspects of the wheelchair, the electroencephalography and the software controlling and connecting the different systems.

2.1 Electroencephalography

“Electroencephalography (EEG) is the physiological method of choice to record the electrical activity generated by the brain from electrodes placed on the scalp surface.” (“What is EEG”, 2016). How these signals arise, where to place the electrodes in order to optimize the information captured and the device used to survey the signals are discussed.

2.1.1 EEG Signals

In order to understand the signals generated by EEG, it is essential to first look at the electrical impulses generated within human tissue. The nervous system is responsible for generating and transmitting these impulses that can be captured. The nervous system is composed out of billions of neurons. Figure 1 shows a typical motor neuron.

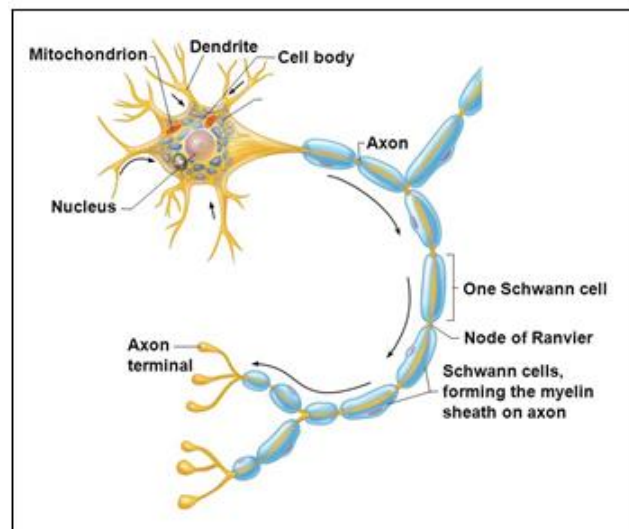


Figure 1: A Typical Motor Neuron (Marieb, 2015)

An inactive neuron is polarized. This means that the outer face of the plasma membrane contains more positive ions than the inner face. The main positive ions inside the cell are potassium while the main positive ions on the outside of the cell are sodium.

Neurons can be stimulated in various ways to become active. Most neurons are excited by neurotransmitter chemicals released by other neurons, while others are stimulated through light, sound, pressure etc. Regardless of the manner of stimulation, the outcome will always be the same. Action potentials (AP) are generated whenever a neuron becomes active and are described in five steps as seen in figure 2:

1. The membrane is in resting state and therefore is polarized. This means that the outside of the membrane is more positive than the inside.
2. A form of stimulus changes the permeability of fragment of the membrane and sodium ions diffuse into the cell. The polarity of the cell starts to change as the outside becomes less positive than the inside.
3. The polarity of the membrane can be entirely reversed if the stimulus is strong enough, which causes an AP to initiated. The membrane is now seen as depolarized.
4. The depolarized membrane causes adjacent membrane fragments to change permeability and sodium ions diffuse freely into the cell. This causes the polarity shift or AP to propagate along the entire length of the membrane.
5. Almost immediately after the sodium ions enter the membrane, does the permeability change again to allow the sodium ions to diffuse out of the cell, restoring the positive charge on the outside and negative charge on the inside of the cell. This returns the cell to a resting state (Marieb, 2015).

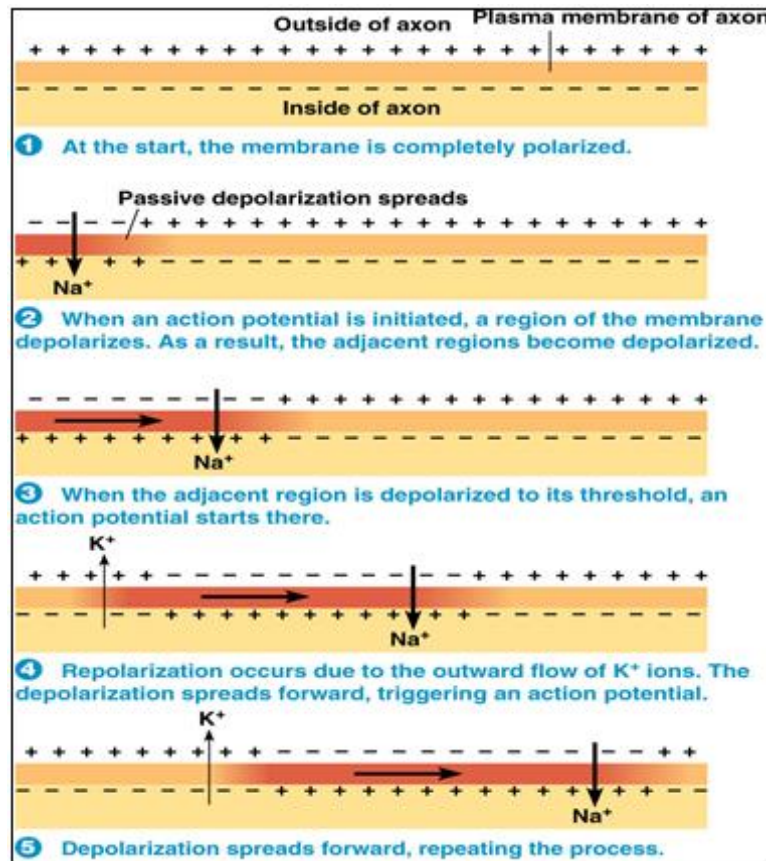


Figure 2: Nerve Impulse propagation (Marieb, 2015)

It is important to know the amplitude of the AP and the time it takes from polarization to depolarization and back to polarization. Figure 3 conveys this data:

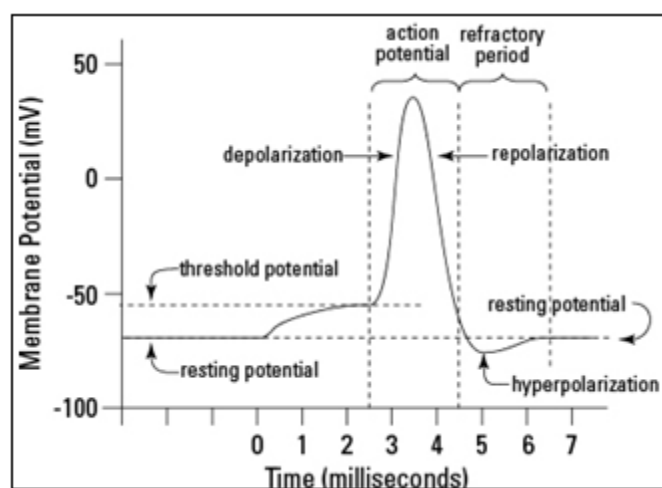


Figure 3: Membrane Potential vs Time (Transmission of Nerve Impulses, 2017)

It is clear to see from figure 3 that the resting potential of a neuron is around -70 millivolt (mV) and the action potential amplitude around 40 mV. EEG signals are electrical activity generated by the synchronized activity of thousands of neurons (“What is EEG”, 2016). Different results are obtained if neurons are not synchronized as shown in figure 4:

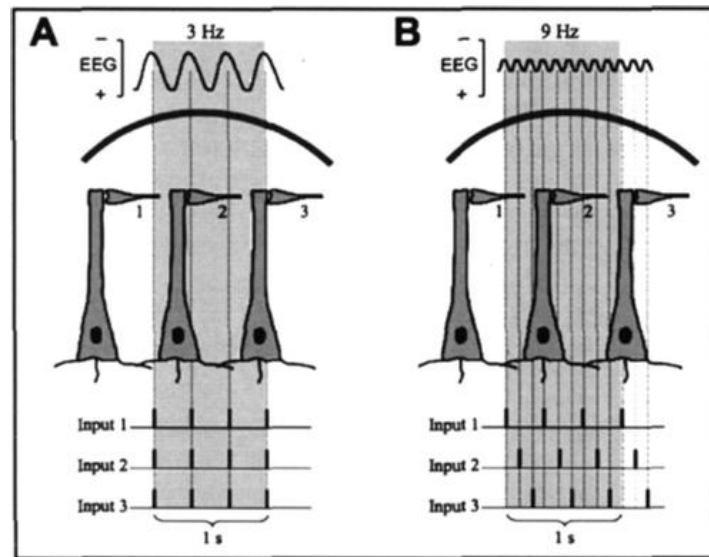







Figure 4: Impacts of Synchronization and desynchronization of Neurons on EEG Signals
(Kirschstein and Köhling, 2009)

Synchronized neurons create EEG signals with low frequency and high amplitude where desynchronized neurons are responsible for low amplitude and high-frequency signals. Different amplitudes and frequencies are categorized in different signal bands and correlate to different mental states as shown in table 1:

Table 1: Frequency Bands' Appearances and Mental States (Garcia et al, 2014)

| Rhythm | Signal appearance | Main behavioral trait |
|-----------------|---|--|
| Gamma 30-100 Hz |  | Represents binding of different populations of neurons |
| Beta 13-30 Hz |  | Usual waking rhythm associated with active thinking and active |
| Alpha 8-13 Hz |  | It is usually found over the occipital regions. Indicates relaxed awareness without attention or |
| Theta 4-8 Hz |  | Theta waves appear as consciousness slips towards drowsiness. Theta increases have |
| Delta 1-4 Hz |  | Primarily associated with deep (slow) wave sleep. |

There are billions of neurons in the human brain and it is impossible to capture the activity of all of them. Even clustering the neurons by the thousands would prove impossible, therefore a protocol is required to maximize the data that can be collected with the minimum amount of electrodes. The electrodes used to capture the data is explained in chapter 2.2.3

As previously stated, the placing of electrodes is vital in order to optimize the EEG signal capturing and assuring the diversity of frequency bands. The most common and effective placement system is called the 10-20 Node Placement and is discussed in the next chapter (Kirschstein and Köhling, 2009).

2.1.2 10-20 Node Placement

The 10-20 placement system is an internationally recognized method used to place electrodes on the scalp to monitor EEG activity. The name 10-20 is used due to electrodes being placed 10% or 20% of the total front–back or right–left distance of the skull. The electrode positions are chosen based on the underlying part of the cerebral cortex (Oostenveld and Praamstra, 2001).

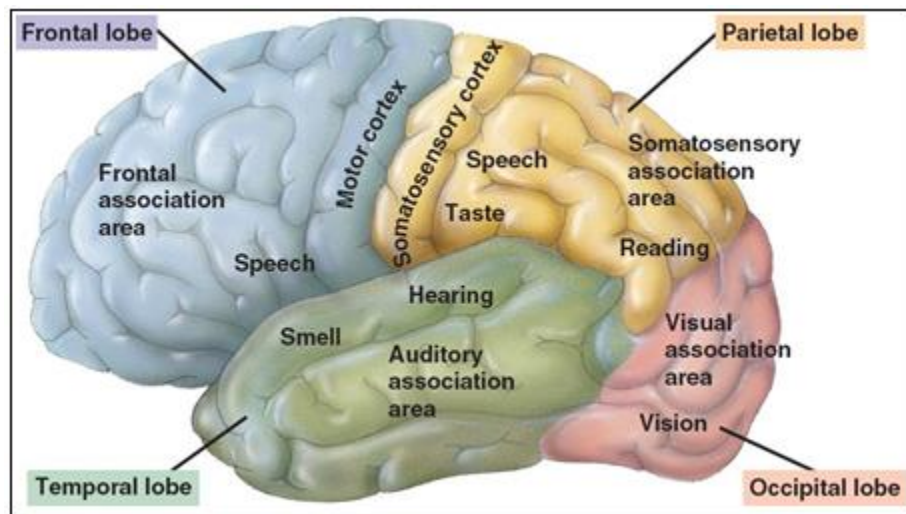


Figure 5: Cerebral Cortex (Sterling, 2017)

The location names are also affiliated with the part of the cerebral cortex on which it is placed. These names are numeronyms which consist of one letter and one number. The letter is associated with the part of the cerebral cortex on which it is placed. The numbers are divided in 3 categories; the numbers on the left hemisphere are uneven, the numbers on the right hemisphere are odd and those on the middle-line are denoted with a z as seen in the figure below:

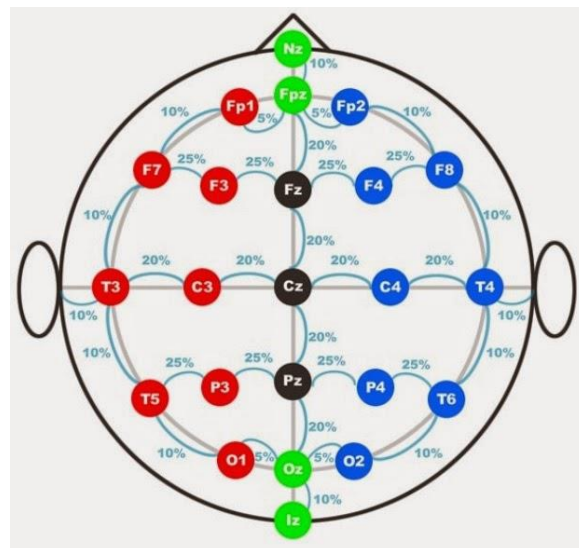


Figure 6: 10-20 Node Placement System (Oostenveld and Praamstra, 2001)

The main reason this system is implemented is to be able to compare results from different subjects. EEG signal bands' intensities are different across various parts of the scalp, thus it is important to be consequent with the electrode placing. The different band intensities are shown below:

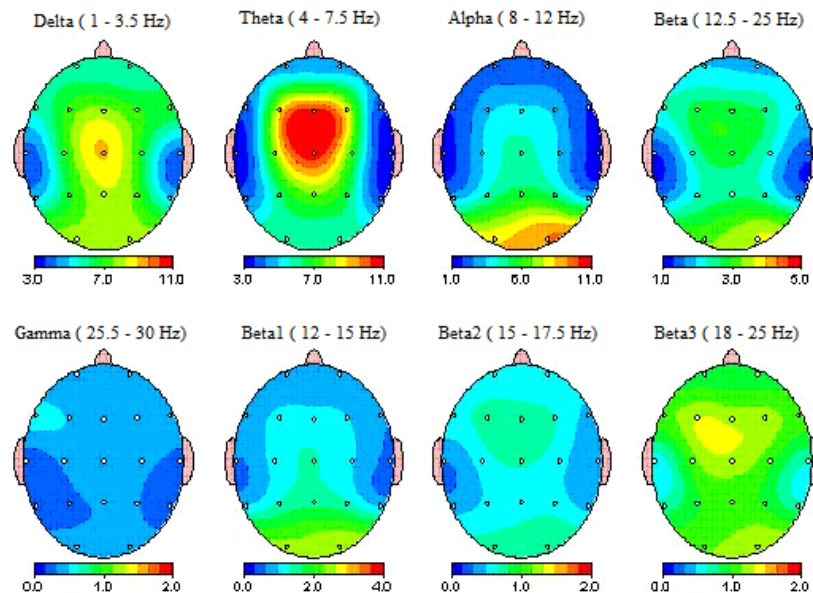


Figure 7: EEG Bands Intensity on the Cerebral Cortex (What is a qEEG?, 2017)

Different techniques are used to utilize the different intensities at various locations on the scalp. These techniques are discussed in chapter 2.2.

2.1.3 Epoc+

The Epoc+ is a non-invasive neuroheadset created by the company Emotiv. There are various phases which neuroheadsets use in order to capture the desired EEG signals.



Figure 8: Epoc+ Neuroheadset (Emotiv, 2014)

The first phase is to extend the raw signals found on the scalp to the headset. This is done by placing electrodes at specific locations (10-20 placement) on the scalp. These electrodes are sponges soaked in saline solutions. According to Clean Water Team (2004), salinity is a measure of the amount of salts in the water. These dissolved ions increase the conductivity of the water. The sponges are connected to highly conductive metals which are connected to an amplifier via wiring.

The Epoc+ uses 14 channels, one electrode for each, and has 4 reference electrodes. The reference electrodes allow the readings to be more accurate as they create a virtual ground for the rest of the sensors.

Once the soaked electrodes are placed on the scalp, the neuro-signals will be transferred to the amplifiers via the metal components and wiring assuming strong contact between the scalp and electrodes. The amplifier increases the current of the signals to allow more accurate readings.

The data is then filtered to eliminate unwanted components such as noise. The amplified signal is then sent to an analog-to-digital converter (ADC). The output of the ADC is then sent via Bluetooth to a USB dongle usually connected to a computer.

The headset has the following specifications:

Table 2 : Epoc+ Specifications (Emotiv, 2014)

| | EEG HEADSET |
|---|---|
| Number of channels | 14 (plus CMS/DRL references, P3/P4 locations) |
| Channel names (International 10-20 locations) | AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8, AF4 |
| Sampling method | Sequential sampling. Single ADC |
| Sampling rate | 128 SPS (2048 Hz internal) |
| Resolution | 14 bits 1 LSB = $0.51\mu\text{V}$ (16 bit ADC, 2 bits instrumental noise floor discarded) |
| Bandwidth | 0.2 - 45Hz, digital notch filters at 50Hz and 60Hz |
| Filtering | Built in digital 5th order Sinc filter |
| Dynamic range (input referred) | $8400\mu\text{V}$ (pp) |
| Coupling mode | AC coupled |
| Connectivity | Proprietary wireless, 2.4GHz band |
| Power | LiPoly |
| Battery life (typical) | 12 hours |
| Impedance Measurement | Real-time contact quality using patented system |

The placing of the electrodes is extremely important in order to optimize signal strength and contact quality. Figure 6 shows the correct placement of the electrodes with the reference signals indicated by the arrows:

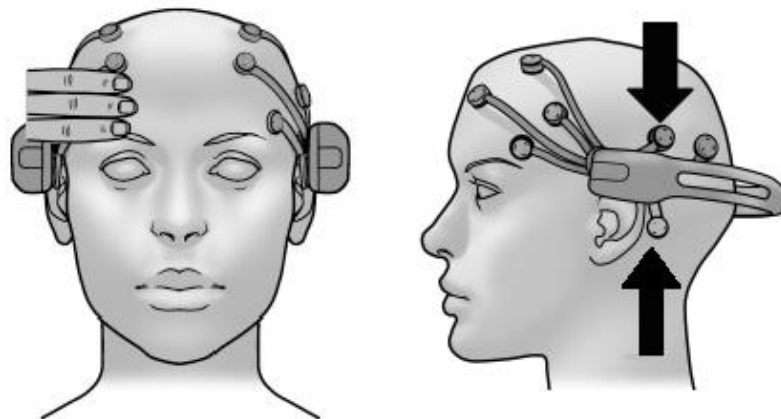


Figure 9: Electrode Placement (Emotiv, 2014)

The Epoc+ is available for commercial use and has a large knowledge base for research purposes.

2.2 Brain-Computer Interface

Brain-Computer Interface (BCI) is the art of combining brain-related activity like EEG signals with existing devices in order to control or monitor an external activity.

2.2.1 P300

P300 is an Event-Related Potential (ERP) which is caused by some form of stimuli, generally of the five senses (auditory, gustatory, olfactory, visual and tactile).

ERPs are electrical activities found on the cerebral surface with a specific amplitude. The P300 component is usually associated with the oddball paradigm experiment (Somani and Shukla, 2014).

P300 potentials are found at the middle-line electrodes (denoted with a 'z') stretching from the frontal lobe to the parietal lobe, increasing its magnitude as it moves closer to the latter. The '300' in P300 is the delay in stimuli. It takes 300 milliseconds (ms) for the ERP to become active on the scalp after the stimulus is experienced. Studies further show that P300 is not generated by the physical attributes of the stimulus, but rather the subject's reaction to it (Somani and Shukla, 2014).

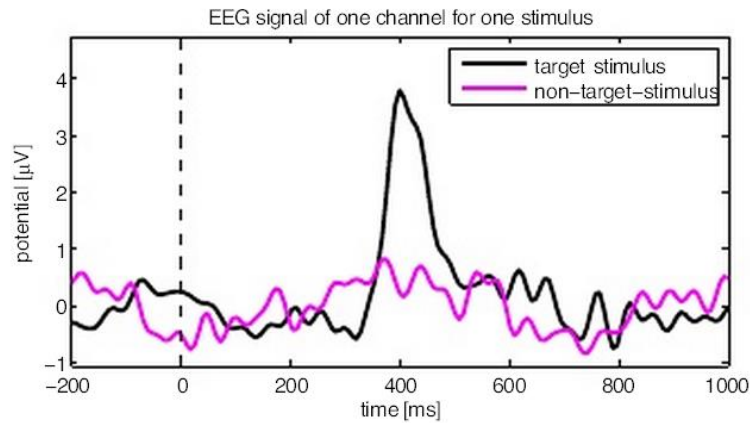


Figure 10: P-300 Stimuli ('Mind-Control' gaming devices leak users' secrets, 2012)

According to Neustadter and Turetsk (2016), P300 waves are decomposed into two different components, P3a and P3b. “The P3a is an earlier frontal response reflecting the engagement of attentional processes, whereas the P3b is a later parietal scalp maximum that is thought to reflect stimulus evaluation associated with environmental representation and memory maintenance” (Neustadter and Turetsk, 2016).

2.2.2 Visually Evoked Potentials

Visually Evoked Potentials (VEP) are evoked potentials that are caused by a visual stimulus. The stimulus are usually easy observable such as flickering lights or alternating patterns. These responses mainly originate from the occipital cortex, which is the area of the brain that is frequently responsible for receiving visual signals as well as interpreting them (Visual Evoked Potential (VEP), 2017).

This VEP has a small amplitude with reference to other potentials and is usually hidden within the EEG signal. These potentials are amplified by repeating the stimulus and using various signal processing techniques to separate it from the background EEG.

The most common VEP contains 4 peaks. The first is a negative peak N1, followed by a positive peak, P1 or P100 because of the timing being 100 ms. Another

negative peak then follows the P100 positive peak, N2, which is almost double the magnitude of N1. The amplitude then rises back to a positive peak which is very small relative to P100 and can be seen as the signal that returns to resting position (Visual Evoked Potential, 2017).

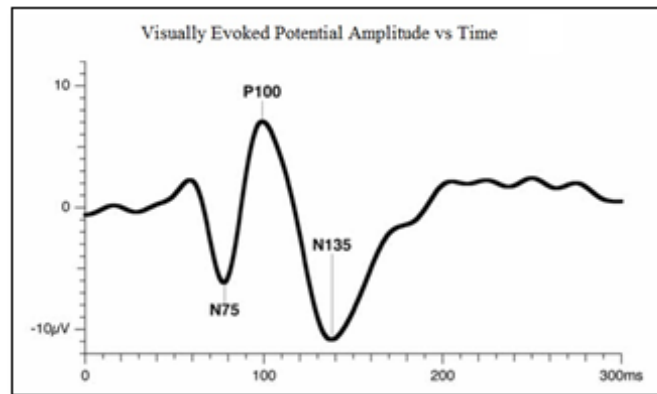


Figure 11: Visually Evoked Potential (Aysha and Nadia, 2016)

2.2.3 Motor Imagery

Motor Imagery in BCI is a standard concept where the user can imagine motor movements to generate induced activity. Motor Imagery is usually recorded over a longer period of time to eliminate the possibility that the captured data could contain a P300 component (Cho et al, 2017).

Motor Imagery is also a type of ERP where the corresponding EEG patterns are purely as a result of a thought. As the name suggests, Motor Imagery components are due to the imagination of a given action.

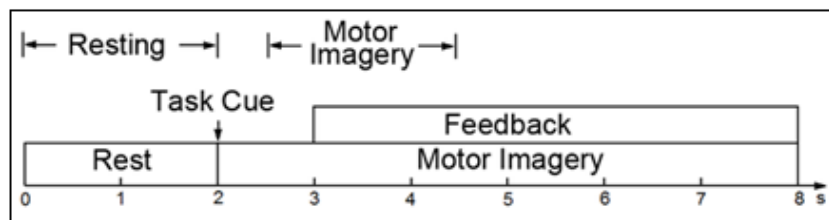


Figure 12: Motor Imagery Delay (Cho et al, 2017).

2.3 DC Motors

For this project, only direct current (DC) motors were researched. DC motors are reliable, accessible and usually cost effective. There are different types of DC motors, however only brushed and brushless motors were researched.

2.3.1 Brushless DC Motors vs Brushed DC Motors

There are 2 main differences between brushless motors and brushed motors. The first is that brushless motors have a magnetic rotor that is surrounded by electromagnets where the brushed motor has an electromagnet system where the rotor is surrounded by permanent magnets. The second factor is that brushed motors have a commutator inside it, which reverse the current direction, where brushless motors' commutators are electronic and reside outside of the motor which tracks the rotor position and activate the coils in a structured way (Brushless -vs- Brushed DC Motors, 2016).

Table 3: Advantages and Disadvantages of Brushless and Brushed DC Motors (Miller, 2010)

| Brushed Motors | | Brushless Motors | |
|---------------------------------------|-------------------------------------|----------------------------------|---|
| Advantages | Disadvantages | Advantages | Disadvantages |
| Cheaper | More Noise | Low maintenance | Expensive |
| No controller need for constant speed | Brushes and commutator can wear out | Superior thermal characteristics | More complicated wiring and controlling |
| Simple wiring | Less efficient | More efficient | |
| | | Higher speeds | |
| | | More power | |
| | | Low noise | |

Brushless motors typically have efficiencies between 85% and 90% where brushes motors have efficiencies between 75% and 80% (Miller, 2010).

2.3.2 Motor Controller

The easiest way to control a brushless motor is with an electronic speed controller (ESC). A standard ESC has 8 wires as shown in the figure below:

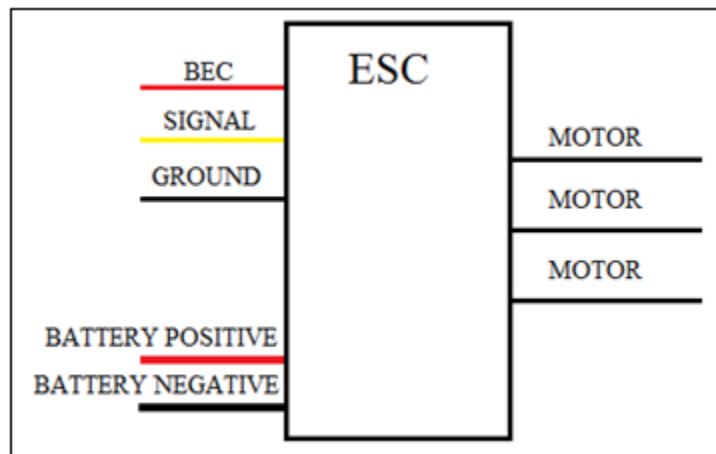


Figure 13: ESC Schematic

The ESC connects to both poles of a battery, one positive and one negative. It also requires a PWM signal and a ground from the same source. Most ESCs have battery eliminator circuits (BEC) that serves as an additional 5-volt input source that can be used to power controllers. An ESC can only be connected to one motor at a time as it only has 3 motor output wires.

ESCs require a PWM signal as input with a frequency of 50 Hertz (Hz) which is a period of 20 milliseconds (ms). The pulse length of the signal must be between 1 ms and 2 ms. At 1 ms the motor does not move and at 2 ms the motor turns at its maximum speed. The pulse width can be changed to a duty cycle (DC) when divided by the total period. The result is that the motor is idling at 5% DC (1 ms pulse) and turning at full speed at 10% DC (2 ms).

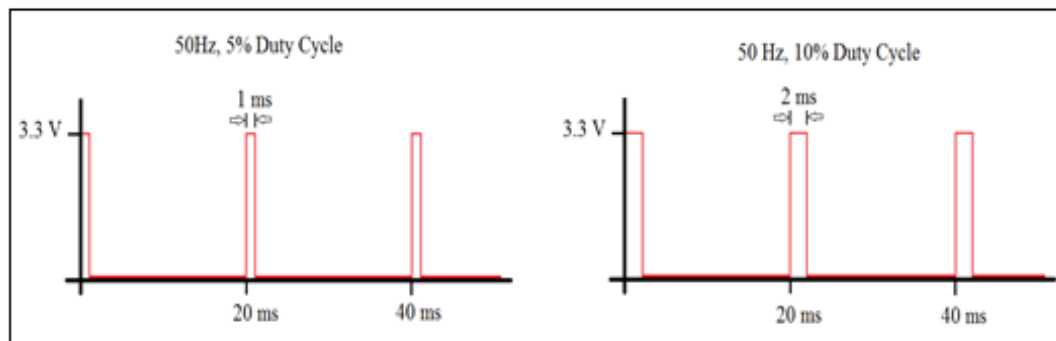


Figure 14: Minimum and Maximum Pulse Width of the PWM Signal

The ESC creates a three-phase alternating current (AC) power output to run the motor. It sends a sequence of AC signals with very low impedance for the rotation. The ESC is also responsible for the commutation. It monitors the back EMF onto the idle phase to synchronize the commutation with the motor's revolutions per minute (RPM).

2.4 System Controller

Various controllers exist today that can effectively do the job required by the project. The main requirement, however, is that the controller must be stand-alone. This rules out most of the commercially available controllers.

The Raspberry Pi is ultimately the only contender that is cost-effective and have the stand-alone property. There exists a great support system for Raspberry Pi's including various libraries that assist when programming.

The official Raspberry Pi power supply can deliver up to 2.5 Amps. This means that with every port and component used on the Raspberry Pi it never exceed this amount. The Raspberry Pi 3 has the following ports:

Table 4: Raspberry Pi 3 Ports

| Port type | Amount |
|--------------------------------------|--------|
| Universal Serial Bus (USB) | 4 |
| Ethernet | 1 |
| Camera | 1 |
| High Definition Multimedia Interface | 1 |
| Mini USB (Power Input) | 1 |
| SD Card | 1 |
| Gate Programmable Input/ Output Pins | 40 |

The Raspberry Pi 3 supports Pulse Width Modulation on numerous pins which can be programmed to the user's specification.

2.5 Pattern Recognition Techniques

Also known as Machine Learning, these algorithms create models which are used to identify known EEG patterns. The considered algorithms are discussed below.

2.5.1 Linear Discriminant Analysis

Linear Discriminant analysis (LDA) is a method used to find a linear combination of features which characterises multiple classes of observations. LDA is used as a linear classifier and in some cases implement dimensionality reduction before classification (Demir and Ozmehmet, 2005).

LDA model the data as a set of multivariate normal distributions with a common covariance matrix for all classes. The LDA classifier uses a Bayes approach. Bayes' theorem of conditional probability is shown below:

$$P(k|x) \propto P(x|k) \propto P(k) \quad (2.1)$$

Where $P(k)$ is the probability of class k , $P(x)$ is the probability of point x , $P(x|k)$ is the conditional probability of point x given class k and $P(k|x)$ is the conditional probability of class k given point x .

The point x is currently being used, thus its probability becomes 1. $P(k)$, the probability of a specific class is generally the same for all classes in the classifier. Most LDA algorithms specify $P(k)$ as 1 divided by the number of classes for example, if there are 5 classes then that probability will be 0.2.

The next step is to calculate $P(x|k)$. First the probability density function (PDF) of point x given class k is calculated:

$$PDF(x|k) = \frac{e^{-\frac{d}{2}}}{(2\pi)^{\frac{p}{2}} \sqrt{|S|}^{\frac{1}{2}}} \quad (2.2)$$

Where d is the squared Mahalanobis distance, S the covariance matrix between the discriminants, observed within that class and p the dimension of the data (Demir and Oz Mehmet, 2005).

The Mahalanobis distance is a measure of the distance between a point and a distribution. It is a generalization of how many standard deviations away a point is from the mean of the class in a multi-dimensional data set.

$$d^2 = (x - \mu)^T S^{-1} (x - \mu) \quad (2.3)$$

Where x is the observation, μ is the mean of the observation class and S the covariance matrix.

Using the PDF will not give the probability of the class containing the observed point. The PDF must be normalized first by using the following equation:

$$P(k|x) = \frac{P(k)PDF(x|k)}{\sum_i^C P(i)PDF(x|i)} \quad (2.4)$$

Where C is the number of classes. This value yields the probability of the observed data to be assigned to class k . The probability of all the classes to assign point x can be calculated and the LDA classifier will assign x to the class with the highest value.

2.5.2 Binary Threshold Classification

The Binary Threshold Classification is a self-taught algorithm used to classify the EEG data. It is not considered machine learning as there are no mathematical models used to classify the data.

In chapter 2.1.1 it was concluded that a neuron or a cluster of neurons can only be in one of its two states. The neurons are either active with an AP across the membrane, or they are passive with a resting potential.

It is possible to create a binary training model based on the amplitude of the EEG data. If the electrode picks up an AP then the channel is seen as a logical '1', if the electrode is at resting membrane potential then the channel is a logical '0'. This is done for all channels to obtain an array of either zeros or ones.

Test data possess the same characteristics as the training data. It is therefore assumed that the test patterns will at some point be identical to the training patterns.

It is now possible to compare the incoming data to the existing arrays to try and find a match in binary. For example, if there is a 2-channel system and the training data was found that some event will happen when both channels are firing (or logical '1').

2-Channels in binary can only yield 4 results: '1','1'; '1','0'; '0','1' and '0','0'. The algorithm keeps monitoring the results until the desired '1','1' presents itself. The algorithm can simply send a signal to the rest of the program to activate some event.

2.7 Available Software

Various software is available that interface well with the Epoc+ that was also created by the Emotiv team. These software include Pure EEG, Emo Composer, Emokey and the research software development kit (SDK).

The research SDK is a series of programs in different programming languages that do the same and more complexed work of Control Panel. The SDK has no user interface and must be implemented in an Integrated Development Environment (IDE). The SDK can fully customise the mental commands and can store and retrieve data from and to cloud servers. Using the SDK requires some experience in software programming.

Pure EEG is used to show raw EEG signals in real time. Different epochs can be saved and loaded to and from files. Pure EEG can also show the Fast Fourier Transform (FFT) and the gyro of the data. This program is primarily used to record data which is used at a later stage.

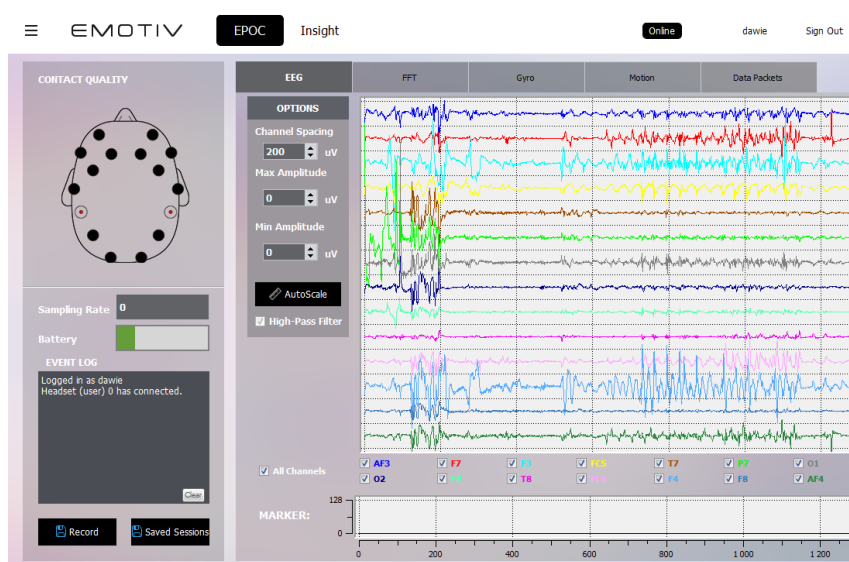


Figure 15: Pure EEG

Emokey is used to create certain actions whenever specific mental commands are found. The required mental commands can be specified as well as the output when such a command is received. Emokey can connect to either Emo Composer for a virtual simulation or to Control Panel which uses real EEG signals.

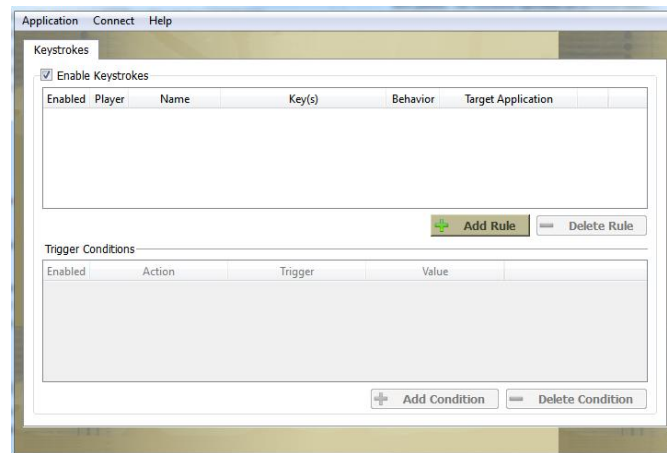


Figure 16: Emokey

Emo Composer is used to simulate mental commands such as thought patterns or physical actions such as winking. Actions can be created by simply clicking on the corresponding buttons, or a pre-defined script can be set up to simulate various commands at specific time intervals.

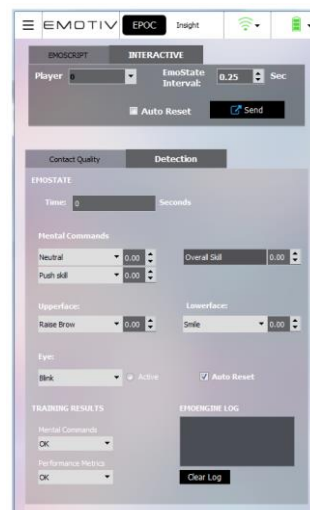


Figure 17: Emo Composer

All these software are commercially available and free except for Pure EEG which requires a monthly subscription.

3. Wheelchair

The wheelchair has a critical role in the project. This chapter thoroughly explains the design and implementation to modify an existing wheelchair into an electric wheelchair.

3.1 Engineering Specifications

Various specifications are set in order to set certain goals for the project. The design specifications are listed in table form to make it easier to compare to results obtained found in chapter 6. The purpose of the specifications is to be able to compare this wheelchair with commercial electric wheelchairs.

Table 5: Engineering Specifications

| Specifications | Value |
|--------------------|--------------------------|
| Nominal Speed | 3 meter per second (m/s) |
| Maximum Carry Mass | 100 Kilogram |
| Maximum Current | 50 Ampere |
| Battery Life | 5 hours |

The design phase can only begin once the specifications are identified.

3.2 Design

This section entails the component choices for the wheelchair as well as the necessary calculations to support the choices.

3.2.1 Wheels

The output shaft of the motors are very small in comparison with the rest of the wheelchair, therefore the wheels need also be small enough to give more leniency towards a gearbox. The front wheels have radius 95 millimetres (mm), therefore it was decided that the driven wheels should be in the same region.

$$T = F \times r \quad (3.1)$$

From equation 3.1: The torque required (T) for the initial movement is proportional to the radius r of the wheel. This creates the requirement that the wheels should be small enough to minimize needed torque while also being able to carry the weight of the entire chair including the user.

There are a finite amount of products available that can handle the total weight while being cost effective. The wheels must also have a bore in the centre where a rod can be inserted and connected to the gearbox.

The choice was the “Guitel Black, Red Castor Wheels”, as seen in figure 1, which is a pneumatic wheel. According to RS Components, pneumatic wheels are designed for shock absorption and rough surfaces (Guitel Black, Red Castor Wheel, 2017).



Figure 18: Guitel Black, Red Castor Wheel (Guitel Black, Red Castor Wheel, 2017)

The wheels have the following specifications:

Table 6: Wheel specification [Guitel Black, Red Castor Wheel, 2017]

| Attribute | Value |
|------------------|-------------------------|
| Load Capacity | 150 decanewton |
| Tyre Type | Pneumatic |
| Hub Material | Polypropylene |
| Features | Low Starting Resistance |
| Hub Bearing Type | Roller Bearing |
| Wheel Diameter | 260 mm |
| Wheel Width | 85 mm |
| Bore Diameter | 25 mm |
| Bore Length | 75 mm |
| Colour | Black, Red |

The wheels can handle up to 150 decanewton (daN), or 1500 Newton (N). This value can be changed to mass by dividing by the gravitational constant 9.81m/s^2 to obtain a maximum load of 152.9 kilograms (Kg). The wheels have a diameter of 260 mm with a 25mm bore diameter, making it ideal for this design. It is also very cost effective.

3.2.2 Motors

The motors are the driving force which directs the wheelchair. It is an essential part of the design. As stated in chapter 2, brushless motors are better suited for this project.

In order to choose a motor, the initial torque required to move the system must be known. However the author obtained two LDPower Brushless Outrunner 2820-

920KV, figure 2, at an extremely low price and decided to do the calculations with said motors before searching for different ones.



Figure 19: LDPOWER Brushless Outrunner 2820-920KV (LDPOWER Brushless Outrunner 2820-902Kv)

The specifications are given below in the table below:

Table 7: Motor specification (LDPOWER Brushless Outrunner 2820-902Kv)

| | |
|------------------|-----------------------|
| Voltage | 11.1~16.8 volts (v) |
| RPM/V | 920 Kv (RPM per volt) |
| Max Power | 640 Watt (W) |
| Max current | 77 Ampere (A) |
| Shaft | 5 mm |
| Suggested ESC | 80 A |
| Motor dimensions | 35 x 40mm |
| Weight: | 137 gram |

Only the maximum power and Kv ratio of the motor is known, as no datasheet could be found. Fortunately, most small DC motors follow the same curve shown below and it was assumed the chosen motors do the same.

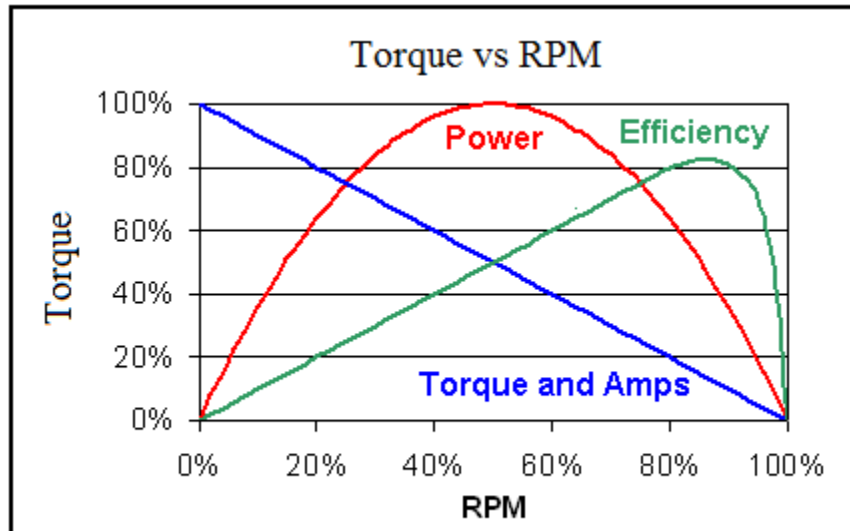


Figure 20: Relationship between Torque, Speed and Power (Motor Torque Specs Minefield)

From figure 12, it is clear to see that the maximum power is present at exactly half the torque and maximum speed. The maximum speed was measured using a tachometer to obtain a speed of 9322 (no load) RPM. The RPM can be converted to radians per second (rad/s) by dividing by 30 and multiplying by Pi. This gives an angular velocity of 976.2 rad/s. The following equation is then used to calculate the stall torque which is the maximum torque the motor can deliver (0 angular velocity):

$$P = T \times w \quad (3.2)$$

Where P is the power of the motor, T the torque and w the angular velocity. Changing the parameters for maximum power yields the following equation:

$$P_{max} = \frac{1}{2} T_{stall} \frac{1}{2} w_{max} \quad (3.3)$$

The maximum power and speed are both known, thus it is possible to solve the stall torque which gives 2.622 Newton meters (Nm). This is a relatively low torque for the amount of weight that is needed to be moved, however the torque can always be increased by the gearbox choice as discussed in the next chapter.

3.2.3 Gearbox

There are 2 requirements of the gearbox; to reduce the speed of the wheels and to increase the torque. For this project, worm gearboxes were the best choice as the wheels could be mounted directly on their output shafts. Also worm gearboxes can withstand high amounts of downward force.

The maximum and minimum stable speed of the motor was measured with a tachometer to get 9233 RPM and 3364 RPM. This is converted to radians per second to give 976.2 rad/s and 352.3 rad/s. The following equation was then implemented to find the required angular velocity of the wheel:

$$V = \omega r \quad (3.4)$$

Where v is the velocity of the wheel, ω the angular velocity of the wheel and r the wheel radius. The speed of the wheel should revolve around 3m/s with a wheel radius of 0.13 m, which yields an angular velocity of 23 rad/s. To find the gear ratio, the following equation is used:

$$GR = \frac{\omega_{motor}}{\omega_{wheel}} \quad (3.5)$$

Substituting the maximum and minimum angular velocities for the motor into the equation 3.6 gives a gear ratio of between 15.3 and 42.4. Higher gear ratios are better for this project as the speed can be adjusted in case the theoretical value does not comply 100% with the design.

The second factor is the torque output of the gearbox. To calculate the total torque required, the following equation is used:

$$T = F \times r \quad (3.6)$$

Where F is the total force exerted on the wheels and r the wheel radius. In most cases, the force due to static friction is greater than that of the dynamic friction force. Thus the higher required torque will reside when the wheel is stationary. Because of the before-mentioned, the force due to drag is negated. Thus the only force the gearbox has to overcome is that caused by friction. The friction force is given by equation 3.7:

$$F = mg\mu \quad (3.7)$$

Where F is the total force, m is the mass of the system, g is the gravitational constant which is 9.81 metres per second per second (m/s/s) and μ the static friction coefficient. The mass of the system is designed to be between 100 and 150 Kg and the static friction coefficient can be found in the following table:

Table 8: Static Friction Coefficients (Friction and Friction Coefficients, 2017)

| Wheel material | Surface Material | Static Friction Coefficient |
|----------------|------------------|-----------------------------|
| Rubber | Asphalt | 0.5 – 0.8 |
| Rubber | Concrete | 0.6 – 0.85 |
| Rubber | Rubber | 1.16 |

The force is calculated for the following surfaces: Rubber on asphalt, rubber on concrete and rubber on rubber. The resulting force as well as the torque required is calculated using equations 3.6 and 3.7:

Table 9: Frictional Force, Torque and Gear ratio values for different masses

| | 100 Kg | 150 Kg |
|--|----------------|----------------|
| Frictional Force (Rubber on rubber) | 1137.96 Newton | 1706.94 Newton |

| | | |
|-------------------------|-----------|-----------|
| Total Torque | 147.93 Nm | 221.9 Nm |
| Torque per Wheel | 73.97 Nm | 110.95 Nm |
| Gear Ratio | 28.21 | 42.32 |

Using equation 3.6 the required torque is calculated only for maximum force. This is the total torque required by both gearboxes. This value is simply bisected to find the torque required by each box individually. The motor's stall torque is already calculated, thus it is possible to find the required gear ratio with equation 3.8:

$$T_{GB} = T_{motor} \times GR \quad (3.8)$$

Where T_{GB} is the torque required by the gearbox, T_{motor} the stall torque of the motor and GR the gear ratio of the gearbox.

It is now possible to choose a gearbox using both the required torque and required speed as factors to select an appropriate gear ratio. The amount of torque the gearbox should be able to accommodate is also known.

The chosen gearbox is the Varvel FRT28 Worm gearbox with a 28:1 gear ratio.



Figure 21: Varvel Worm Gearbox (Worm gearbox / right-angle, 2017)

The gearbox has a ratio of 28:1 which fits into the specification. It also has a maximum torque of 691 Nm which is more than enough (Worm gearbox / right-angle, 2017). Worm gearboxes are designed to carry a big portion of the total weight, reducing the force on the motors and shafts. These gearboxes' input and output shafts create a 90-degree angle with each other allowing more freedom to mount both the gearboxes and motors onto the wheelchair.

3.2.4 Battery

The battery used in this project is Victron Energy 12V 60Ah-55Ah absorbed glass mat (AGM) Deep Cycle Battery. The battery has a rated 10-hour discharge at 25C. This means that the battery can effectively provide 5.5 – 6 Amps per hour for 10 hours straight. This is considered safe since the motors will not pull a high amount of current (Critical Power, 2009).



Figure 22: Victron Energy 12V 60-Ah-55Ah (Critical Power, 2009).

The battery can, when needed, provide 25 times its normal discharge rate. This will cause the battery capacity to deplete quickly. The main causes for higher discharge rates will be a change of terrain (with a higher friction), uphill traveling or speed increments.

3.2.5 Additional Designs

For this project, 5 different components were designed and drawn (refer to Appendix D). The parts in order are the Bar, Bracket, Key, Coupling and Rod.

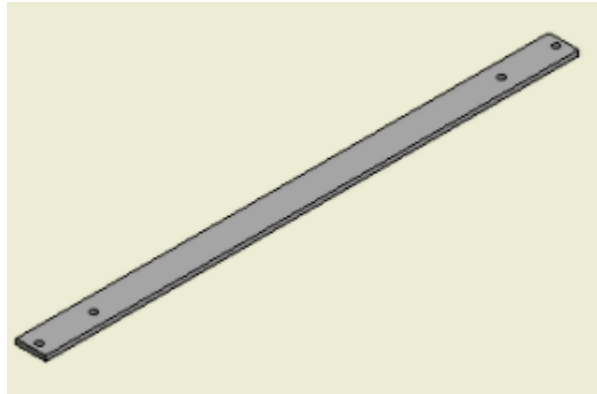


Figure 23: Bar Design

The bar is designed to run across the wheelchair and allow the gearbox to be mounted on it. The bar must be strong enough to carry the weight of both gearboxes and motors without too much bending over a length of 430 mm. The bar has 4 holes in total, 2 at each end point.

The second part is the bracket which attaches the motor to the gearbox.

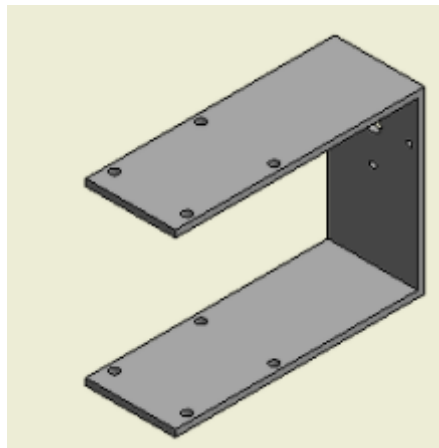


Figure 24: Bracket Design

The bracket is designed to fit onto both the upper and lower part of the gearbox to minimize the bending moment on the motors. The bracket has 4 holes at the top as well as 4 at the bottom and 4 slightly smaller holes on the side for the motor mounting.

The key is used to attach the motor to the gearbox.

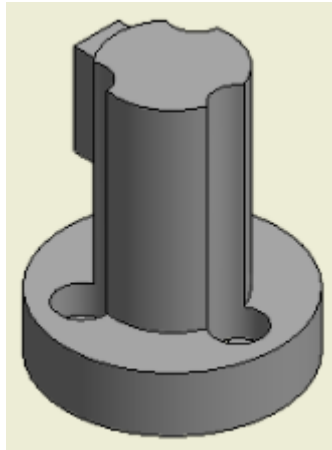


Figure 25: Key Design

The key is designed to attach onto the motors' top side and fit into the coupling on the other side. The key has 3 holes on the base which matches that of the motor.

The following couplings came with the gearbox:



Figure 26: Coupling Top View

The couplings are manufactured to fit into the gearbox input slot. They have a hole in the centre with a slot fit for a key as shown in figure 25.

The last custom item is the rod.

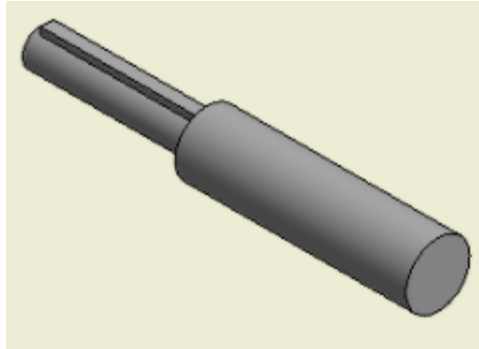


Figure 27: Rod Design

The rod has two different size cylinders, one for the wheel and one for the gearbox. The part that fits into the gearbox also has the appropriate key size.

3.3 Wheelchair Assembly

The assembly consists of the following parts:

- Manual Wheelchair
- 2x LDP Brushless Outrunner motors
- 2x Varvel Worm 28:1 Gearboxes
- 2x Guitel Black and Red Castor Wheels
- Victron 12V 60-55 Ah battery
- 2x Bars
- 2x Brackets
- 2x Keys
- 2x Couplings
- 2x Rods
- 4x 25 mm Circlips

- 4x 14 mm Circlips

The brackets and bars are mounted at the bottom of the wheelchair as shown in the following figure:

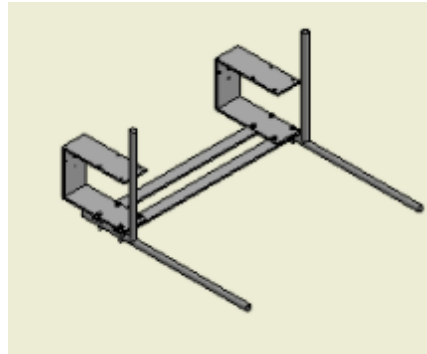


Figure 28: Assembly Design

The wheelchair used to design this assembly has two 120 mm unused horizontal rods towards the back-end of the frame. The rod diameters are 12.7 mm and the rods are placed 430 mm away from each other.

The motors are mounted through the holes on the side of the bracket. The keys mount directly on the motors which fits into the couplings. The couplings are inserted into the gearbox. The gearboxes are then mounted on the bottom and top of the brackets.

The next step is to insert the rods through the gearboxes with the correct key positioning. The rod is further inserted into the wheel. Circlips are used to prevent horizontal movement on the rods.

4. Controller Hardware

The Raspberry Pi was chosen to control the system mainly due to its independence. The controller receives commands from the classification algorithms via the laptop

and sends the appropriate PWM signal to the speed controllers to steer the wheelchair.

4.1 Raspberry Pi

The controller chosen for this system is the Raspberry Pi 3b. The main function of the controller is to receive commands from the laptop and send appropriate PWM signals to the ESCs.



Figure 29: Raspberry Pi 3B (Raspberry Pi 3 Model B, 2017)

The commands are transferred from the laptop to the Raspberry Pi via standard Ethernet connection at 100 Megabits per second (Mb/s).

The Raspberry Pi is powered by a 5200 milliamp hour (mAh) power bank battery which can deliver up to 2.1 Amps at any given time. The Raspberry Pi model B uses roughly between 700 mA and 1000 mA (Raspberrypi.org, 2017) which deems the battery a safe choice.

The Raspberry Pi has 40 pins, of which only 4 are used for this project. 2 pins are used for ground and the other 2 are for Pulse-Width Modulation signals (PWM). These signals have a frequency of 50 Hz which gives a period of 20 ms. The PWM signals also have an amplitude of 3.3 V and pulse width varying from 1 ms to 2 ms depending on the required speed of the motor as discussed in chapter 4.2.

4.2 Electronic Speed Controllers

Electronic Speed Controllers (ESC) are used to regulate the speed of DC motors. ESCs and DC motors are incorporated together to ensure that the motor does not draw more current than the ESC can handle.

The chosen ESC is the Hobby King SS series 50-60Amp. The maximum current of the ESC is less than the maximum current of the motor which could cause problems, however the total current needed should not exceed the ESC's maximum as 2 motors are used simultaneously.



Figure 30: Hobby King SS Series 50-60A ESC (Hobbyking™ SS Series 50-60A ESC (Opto only), 2016)

This ESC includes a battery eliminator circuit (BEC) which generates an extra power source which has a potential of 5 volts. The Raspberry Pi uses an external battery, thus the additional power supply will not be necessary and is discarded from the project.

As discussed in chapter 3, the speed at which the motor must turn to reach 3 m/s is 646.15 radians per second as derived from equation 3.5. The formula to calculate the desired pulse width in order to cause the motor to turn at the required speed is given below:

$$Pulsewidth = \left(\frac{DesiredRPM}{\frac{MaxRPM}{100}} \times 10ms \right) + 1 \quad (4.2)$$

The maximum and desired RPM are both known from chapter 3 thus the pulse width is simply calculated to give 1.658 ms.

This ESC can also program the motor to some extent. To program the ESC, switch the PWM output to its highest pulse width (2 ms) before connecting the battery and wait for 6 seconds. A special tone starts to play which signals that programming mode has been entered. The ESC then plays the tones for each category on the list seen in the table below. Once the desired item's tone is played, switch the PWM output to its lowest pulse width (1 ms). The ESC will play a different tune which means the chosen item in the menu has been selected and applied.

Table 10: ESC Program Options (User instruction for RC aircraft ESC, 2017)

| | | | |
|---|----------------------------------|--------------------|---------------------|
| Program: (5 types Prompt tone are as following) | | | |
| A=BEEP- short sound | | | |
| B=BEEP-BEEP-BEEP- 3 continuous sound | | | |
| C=~BEEP gradient sound+BEEP | | | |
| D=BEEP\ low sound | | | |
| E=BEEP-- long sound | | | |
| music 1 | throttle----throttle calibration | | A-A-A-A |
| music 2 | brake | | B-B-B-B |
| music 3 | Battery type | Ni-MH | C-C-C-C |
| music 4 | | Li-Po | D-D-D-D |
| music 5 | Low-voltage protection threshold | low (2.8V) | E-E-E-E |
| music 6 | | medium (3.0V) | AA-AA-AA-AA |
| music 7 | | high (3.2V) | BB-BB-BB-BB |
| music 8 | Restore factory default | | CC-CC-CC-CC |
| music 9 | timing | automatic | DD-DD-DD-DD |
| music 10 | | low (7° -22°) | EE-EE-EE-EE |
| music 11 | | high (22° -30°) | AAA-AAA-AAA-AAA |
| music 12 | motor start | very soft | BBB-BBB-BBB-BBB |
| music 13 | | soft | CCC-CCC-CCC-CCC |
| music 14 | | Acceleration start | DDD-DDD-DDD-DDD |
| music 15 | helicopter mode | OFF | EEE-EEE-EEE-EEE |
| music 16 | | helicopter 1 (5S) | AAAA-AAAA-AAAA-AAAA |
| music 17 | | helicopter 2 (10S) | BBBB-BBBB-BBBB-BBBB |
| music 18 | motor rotation direction | | CCCC-CCCC-CCCC-CCCC |
| music 19 | PWM motor frequency | 8 kHz | DDDD-DDDD-DDDD-DDDD |
| music 20 | | 16 kHz | EEEE-EEEE-EEEE-EEEE |
| music 21 | low-voltage protection mode | power reduction | AD-AD-AD-AD |
| music 22 | | Cut off output | AE-AE-AE-AE |

The throttle is calibrated to give minimum throttle at 1 ms pulse width and maximum throttle at 2 ms pulse width. The timing is set to high and the PWM frequency to 16-kilo hertz, because of the rule of thumb that higher poles need higher frequencies. The only other change is the motor start changed to very soft to slow down the motor acceleration to protect the gearbox. By default, the brake is disabled, the low-voltage protection used is power reduction, helicopter mode is disabled and the low-voltage protection threshold is set to medium or 3 v (User instruction for RC aircraft ESC, 2017).

4.3 Central Processing Unit

A laptop is used as a link between the Raspberry Pi and the Epoc+. It also does all the computational work with regards to the machine learning algorithms.

The laptop used is a Samsung brand with an Intel Core i5-2467M CPU @ 1.60 Giga Hertz (GHz) and 4 Gigabyte (GB) installed memory (random access memory). The laptop runs on Windows 7 Home Edition. The only ports that are used in this project is one USB port and one Ethernet port.

4.4 Epoc+

The Epoc+ was chosen for the neuroheadset as it was the only choice. The Epoc+ has 14 active electrodes with 2 of them providing reference signals.

The headset is placed on the user's head and the data is transferred to the laptop. The Epoc+ sends the data via Bluetooth to the dongle which is plugged into the USB port of the laptop. The data is sent at 25 Mb/s.

4.5 System Connections

The components needed to create this circuit are the following:

- Raspberry Pi 3b

- Laptop
- 2x 50-60A Hobbyking ESCs
- 5V – 5200mah Power Bank
- Epoc+ Neuroheadset
- Network Cable
- Wiring

Only after all the designs and component choices are complete, can the system be assembled. The complete hardware connections can be seen below:

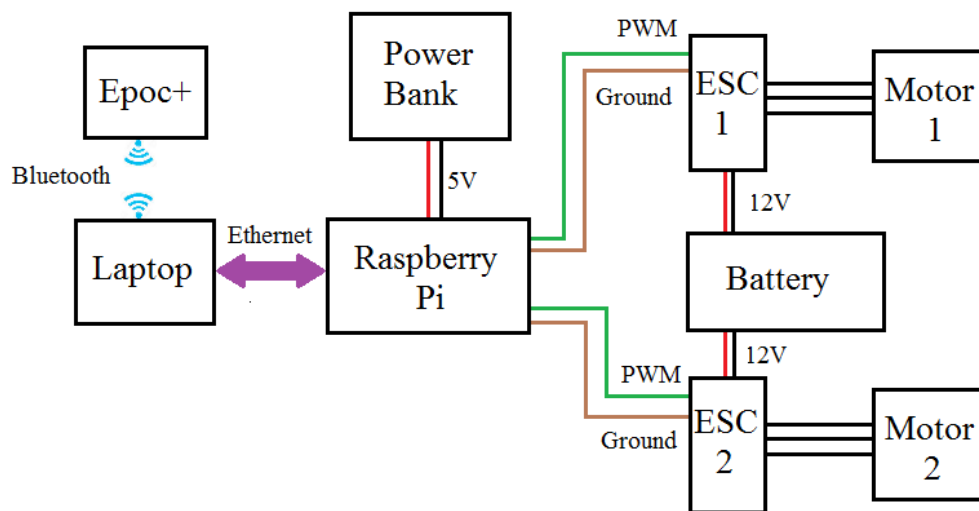


Figure 31: System Diagram

The neuroheadset is placed on the user's head and the USB dongle is plugged into the laptop. The headset sends the EEG signals to the laptop via Bluetooth. The laptop then runs the data through pattern recognition algorithms and obtains the highest likelihood result. The laptop then sends a code corresponding to the movement the wheelchair must follow from the EEG data to the Raspberry Pi through an Ethernet connection. The Raspberry Pi which is powered by a power bank must then adjust the PWM signals (if necessary) and send them to the ESCs through copper wires. The ESCs which is powered by a 12 v deep cycle battery

alters the signals they send to the motors in order to adjust the motor speed (or in some cases keep it as it is).

5. Controller Software

Software required to control the wheelchair is found on both the laptop and the Raspberry Pi. The program on the laptop does the pattern recognition and sends commands via the Ethernet port to the Raspberry Pi which controls the wheelchair motors. Python programming language is used for both the laptop's machine learning algorithm and for the Raspberry Pi's motor control.

5.1 Serial Communication

The communication system used in the project is called Ethernet connection. This requires a Local Area Connection (LAN) Cable between the laptop and the Raspberry Pi.

The Raspberry Pi is set up to be the server while the laptop is the client. Once the server is set up, it waits for activity on the Ethernet port. Only once the laptop sends specific data to the Raspberry Pi, will it react. The following code is a similar snippet to that found on the Raspberry Pi. The only difference is the Raspberry Pi continues to listen after a command has been received.

```
import socket                                     #Use the socket library

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #Create a socket with default address family and sock stream socket type
s.bind(('192.168.0.21', 5005))                    #Sets up socket to be a server with given IP address and port number
s.listen(1)                                       #Waits until data is received
conn, addr = s.accept()                          #Accepts the data from the sent address
data = conn.recv(20)                             #Gets the command in readable format
```

Figure 32: Server Ethernet Setup

The laptop is the client in this instance. The client cannot receive data from the server, it can only send it to the server. If the classification algorithms conclude that change of state is required, then the laptop will send the corresponding letter to the Raspberry Pi. The following image shows the methodology of a socket connection used in this project:

```

import socket                                     #Use the socket library

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #Create a socket with default address family and sock stream socket type
s.connect(('192.168.0.21',5005))                  #Allows socket to connect to the server with given IP address and port number
s.send('w')                                       #Send command for forward movement
data = s.recv(1024)                             #Check to see if data was successfully sent

```

Figure 33: Client Ethernet Setup

The client can only send data to the server, and the server can only receive data from the client.

5.2 Raspberry Pi Code

As mentioned above, both the Raspberry Pi and the laptop require the Ethernet connection code to run during the entire process. The Raspberry Pi code have 2 main functions part from the serial data communication.

The first part is to calibrate the motors. This is done by sending the appropriate pulse width at the correct time to program the motors as shown in chapter 4.2. For this, pins 12 and 33 are used as they are PWM output pins with pins 6 and 9 as the ground sources. It is important to notice that the chosen pins for PWM signals are not on the same channel.

| Raspberry Pi 3 Model B (J8 Header) | | | | | |
|------------------------------------|----------------------|--|--|----------------------|------|
| GPIO | NAME | | | NAME | GPIO |
| | 3.3 VDC Power | | | 5.0 VDC Power | |
| 8 | GPIO 8 SDA1 (I2C) | | | 5.0 VDC Power | |
| 9 | GPIO 9 SCL1 (I2C) | | | Ground | |
| 7 | GPIO 7 GPCLK0 | | | GPIO 15 TXD (UART) | 15 |
| | Ground | | | GPIO 16 RXD (UART) | 16 |
| 0 | GPIO 0 | | | GPIO 1 PCM_CLKIN0 | 1 |
| 2 | GPIO 2 | | | Ground | |
| 3 | GPIO 3 | | | GPIO 4 | 4 |
| | 3.3 VDC Power | | | GPIO 5 | 5 |
| 12 | GPIO 12 MOSI (SPI) | | | Ground | |
| 13 | GPIO 13 MISO (SPI) | | | GPIO 6 | 6 |
| 14 | GPIO 14 SCLK (SPI) | | | GPIO 10 CSD0 (SPI) | 10 |
| | Ground | | | GPIO 11 CS1 (SPI) | 11 |
| 30 | SDA0 (I2C ID EEPROM) | | | SCL0 (I2C ID EEPROM) | 31 |
| 21 | GPIO 21 GPCLK1 | | | Ground | |
| 22 | GPIO 22 GPCLK2 | | | GPIO 26 PWM0 | 26 |
| 23 | GPIO 23 PWM1 | | | Ground | |
| 24 | GPIO 24 PCM_FSIN0 | | | GPIO 27 | 27 |
| 25 | GPIO 25 | | | GPIO 28 PCM_DIN | 28 |
| | Ground | | | GPIO 29 PCM_DOUT | 29 |

Figure 34: Raspberry Pi 3b Pin Layout (. Pin Numbering - Raspberry Pi 3 Model B, 2016)

The calibration and ESC programming algorithms are only used once as the ESC has an internal memory and can save the values it was last assigned.

The second part of the Raspberry Pi code is to change the PWM signals so that the motors rotate in the direction required. The algorithm is set up so that the system constantly listens to the Ethernet connection for commands from the laptop. At this point there are only 6 different states: Forward, Stop, Left, Right, Idle and Exit. The Raspberry Pi will only adjust the PWM signals if it receives a command to change its state. The 6 states are described in the table below:

Table 11: Raspberry Pi Commands

| State | Command | Action |
|---------|---------|---|
| Forward | 'w' | Both motors are turned on and pulse width of signals increase to reach desired speed over course of 5 seconds |
| Stop | 's' | Both motors are shut down starting at current speed and decreasing to zero speed over 1 second |
| Left | 'a' | Left motor speed is reduced to 0 for 0.5 second |
| Right | 'd' | Right motor speed is reduced to 0 for 0.5 second |
| Exit | 'q' | Exit the program entirely and disables the output pins |
| Idle | 'I' | Continue with previous command, acts as delay buffer |

The idle state was introduced due to the fact that it is the most common found state. The algorithm classifies every time new data is introduced which, according to the Epoc+ sample rate, is 128 samples every second. The algorithm should classify the

incoming data after a primary state (Forward, Stop, Left, and Right) is completed to the idle state to allow the data to settle before a next state is assigned.

The following figures are obtained from an oscilloscope which show the PWM outputs of the minimum and maximum pulse width of the Raspberry Pi:

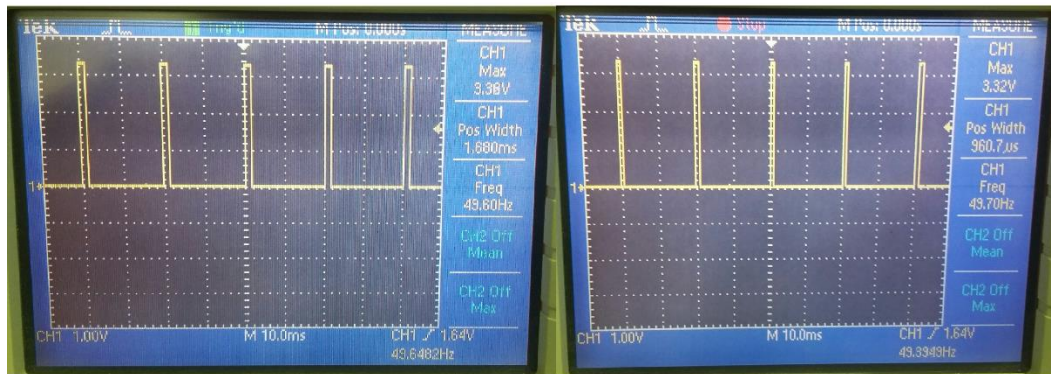


Figure 35: Maximum Pulse Width Output (Left) and Minimum Pulse Width Output (Right) at Pin 12 of the Raspberry Pi

As specified, the measure amplitudes are 3.3 V, the frequency 50 Hz and the pulse width varying from 1 ms to 1.68 ms.

The following is a snippet from the Raspberry Pi code to show how the motors' speed are changed in case of the forward command:

```
#Increase both motors' speed
def forward():
    #reduce and increase speed over 0.5 seconds
    for i in range(0,10):
        if RIGHT_SPEED < 8:
            RIGHT_SPEED += 0.3
            pwm_right.ChangeDutyCycle(RIGHT_SPEED)

        if LEFT_SPEED < 8:
            LEFT_SPEED += 0.3
            pwm_left.ChangeDutyCycle(LEFT_SPEED)

    time.wait(0.05)
```

Figure 36: Raspberry Pi Code in Case of Forward Movement

Similarly are there methods for each of the 4 movement commands.

5.3 Acquired Data

Data acquisition is an important concept. There are 2 types of data that need to be addressed, training and test data.

5.3.1 Training Data

The training data was acquired through a program called Pure EEG. Pure EEG reads the users EEG signals continuously with the option of exporting the data to csv files.

The training data was obtained by consciously thinking about the specific command required and recording the data for 5 seconds around the potential. The data is taken for 5 second epochs to eliminate the P300 component. This is repeated 6 times for each command to increase the accuracy.

There are five command stages that were recorded (6 times each): Forward, Stop, Right, Left and Idle.

The training data was acquired prior to the activation of the project. The acquisition of training data is also a once off procedure.

5.3.2 Test Data

The test data is obtained by continuously reading in the EEG data from the Epoch+ while the system is running. The data is dynamically acquired and stored in buffers, while waiting to be processed and classified. The buffers store 100 values per channel.

The buffers will be filled with new data after the data is classified. All the old data will move up one element in the buffer per new line of data recorded. The old data will simply be discarded once new data is available.

5.4 Signal Processing

The signal is necessary to partially normalize the data. This step is the same for both the test and training data. The entire channel's average is calculated and subtracted from each individual value.

```
#subtract the mean
for i in range(0,14):
    means.append(np.mean(data[i,:]))
for i in range(0,14):
    data[i,:] -=means[i]
```

Figure 37: Code Snippet for Subtracting Means

This prevents the use of large values in the data and creates a more attractive output. It is also easier to discover when a channel has fired or not (logical '1' or '0'). The new values also indicate the voltage level of the channels. A typical output would look as follow:

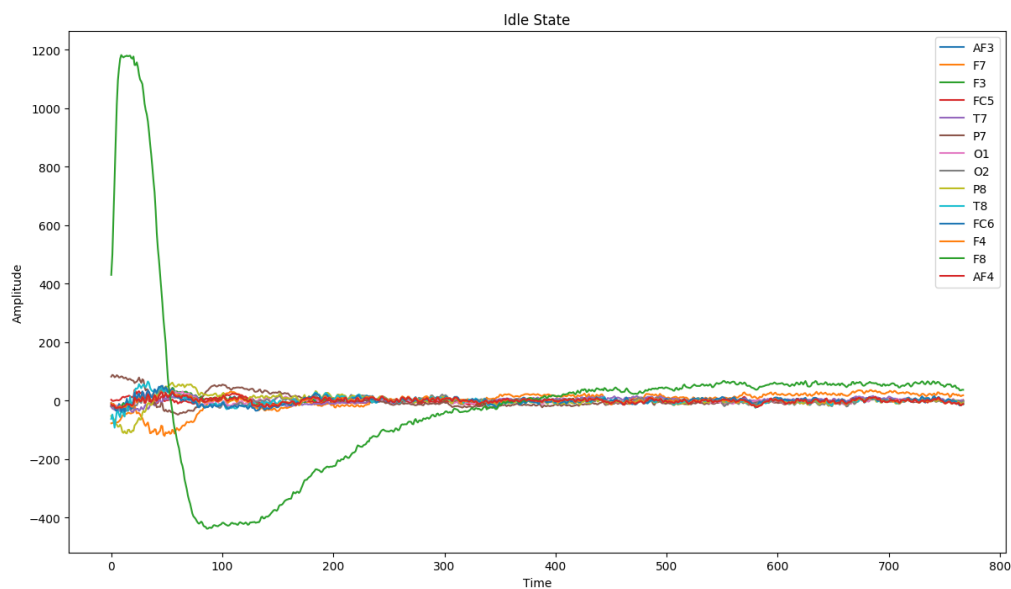


Figure 38: EEG Signals of Idle State

These graphs exist for all 5 states and can be found in Appendix F. The next step is to create classifiers using this data.

5.5 Classification Software

The two classification methods described in chapter 2 are discussed.

5.5.1 Binary Classification

The Binary classification was implemented in the system. The software can be divided into different steps.

The first step is to identify the training data and transform it into a binary array. The cut-off threshold for a binary '1' is below the value 50. The absolute value of the signal was taken to reduce the probability that more than one state will occur. This data is normalized thus the values are lower than the raw outputs. The training data was averaged over the 6 different attempts to get the most accurate solution.

The results for the training data was obtained:

Table 12: Binary Values of Training Data

| | Forward | Stop | Left | Right |
|------------|----------------|-------------|-------------|--------------|
| AF3 | 0 | 1 | 1 | 1 |
| F7 | 1 | 1 | 1 | 0 |
| F3 | 0 | 1 | 1 | 1 |
| FC5 | 1 | 0 | 1 | 1 |
| T7 | 1 | 1 | 1 | 0 |
| P7 | 0 | 0 | 1 | 0 |
| O1 | 0 | 0 | 0 | 1 |
| O2 | 0 | 0 | 0 | 0 |
| P8 | 0 | 0 | 0 | 0 |

| | | | | |
|------------|---|---|---|---|
| T8 | 1 | 1 | 1 | 0 |
| FC6 | 1 | 1 | 0 | 0 |
| F4 | 0 | 1 | 0 | 1 |
| F8 | 1 | 1 | 1 | 1 |
| AF4 | 0 | 1 | 0 | 0 |

The incoming data is then compared to all 4 classes. For each channel the observed data is similar to the classifiers, the state will gain 1 score. The state with the highest score at the end of the classifier will be the most likely next state. In the rare case of more than one state having the same high score, then the Idle state will continue until one of the before mentioned states surpasses the other.

5.5.2 LDA

The LDA was also implemented to have a comparison with the above mentioned classifier to be able to weigh the two classifiers against each other. The LDA used was that of the Scikit-learn libraries.

The data is first normalized and then given labels 0 to 4 each belonging to a state. The model is then set up for 750 different observations on each class. The Scikit-learn makes it easier to implement the classifier. The only code needed to classify an entire observation, once the model has been set up, is:

```
x = lda.predict(np.array([data[:,749]]))
```

This will return the most likely class label that the data should belong to. The LDA was set up so it can only classify every 0.5 seconds to give the motors time to finish turning the wheelchair.

5.6 Program Flow

The program flow is shown in the figure below:

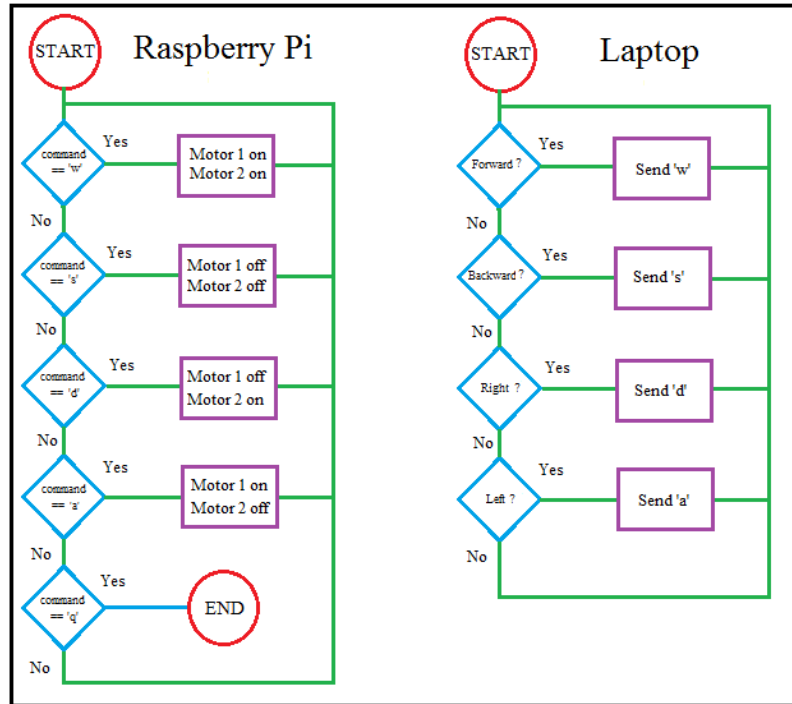


Figure 39: Program Flow for Raspberry Pi and Laptop

The laptop in connection with the neuro-headset keeps running the classification algorithm until a match is found. If a known pattern is recognized then the laptop will send the appropriate command to the Raspberry Pi via the Ethernet port. If no known pattern is found then the program will continue until new data is received on which it will execute the classification algorithm.

The Raspberry Pi waits until it receives data from the serial port. If the data does not match one of the 5 commands then it will continue to listen. If the data is in fact a command issued by the laptop, then the Raspberry Pi will adjust the PWM signals as shown in the figure above.

6. Results

The hardware and software of the system was tested to compare with the specifications provided in chapter 3.

6.1 Battery Life

An important specification is the duration of the batteries. This result was recorded for the wheelchair, Raspberry Pi and the laptop.

The wheelchair's battery was tested by only using the no-load motor values. The wheelchair's battery was active for a measured 3 hours before the voltage dropped below an acceptable value. The battery was not charged to its fullest before the test was conducted.

The power bank which powers the Raspberry Pi lasted 4 hours and 30 mins before the system was switched off. This is an acceptable result seeing as this battery must outlive that of the wheelchair's, irrelevant by what margin.

The laptop's battery lasted 6 hours before it was shut down. Once again this result is favourable as it outlives that of the wheelchair.

6.2 Classification Algorithm

The classification algorithm implemented is the binary classification algorithm and the following results were obtained.

6.2.1 Timing

The algorithms was set up so that only one classification can exist every half second. The delay is used to give more time to the hardware aspect. Also while the delay is active, will the system still pass data through. This means that after the

delay, will the classification algorithms be introduced to new data. The neuro-headset has a sample rate of 128 Hz which means that in the half second delay, the algorithms will have 64 new observations to process.

6.2.2 Accuracy

Testing accuracy of dynamic data proves a difficult task. The data must be monitored while the headset is in use.

The system was tested using the four known states. The idea of the testing was to try and mimic the same sequence as shown below. The sequence tables are read first from left to right then top to bottom.

Table 13: Sequence of Classification Algorithm testing

| Sequence 1 | | | |
|-------------------|---------|---------|---------|
| Left | Left | Left | Left |
| Right | Right | Right | Right |
| Forward | Forward | Forward | Forward |
| Stop | Stop | Stop | Stop |
| Sequence 2 | | | |
| Left | Right | Forward | Stop |
| Left | Right | Forward | Stop |
| Left | Right | Forward | Stop |
| Left | Right | Forward | Stop |

The following results were obtained in the form of a confusion matrix. The LDA results are given first then the binary classifier. The output of a confusion matrix shows the difference in predicted labels versus the actual labels. The predicted labels are in the horizontal position where the actual labels are on the vertical axis.

Table 14: Confusion Matrix for LDA

| Sequence 1 | | | | |
|-------------------|-------------|--------------|----------------|-------------|
| | Left | Right | Forward | Stop |
| Left | 4 | 0 | 0 | 0 |
| Right | 0 | 4 | 0 | 0 |
| Forward | 2 | 0 | 2 | 0 |
| Stop | 1 | 1 | 1 | 1 |
| Sequence 2 | | | | |
| | Left | Right | Forward | Stop |
| Left | 2 | 0 | 1 | 1 |
| Right | 1 | 2 | 1 | 0 |
| Forward | 0 | 1 | 2 | 1 |
| Stop | 0 | 0 | 0 | 4 |

For sequence 1 there is an accuracy of 68.75% and sequence 2 has an accuracy of 62.5%. This is a relatively low accuracy. The LDA has better accuracy when there are reoccurring patterns.

The confusion matrix of the Binary classification is given:

Table 15: Confusion Matrix for Binary Classifier

| Sequence 1 | | | | |
|-------------------|-------------|--------------|----------------|-------------|
| | Left | Right | Forward | Stop |
| Left | 2 | 0 | 2 | 0 |
| Right | 1 | 2 | 1 | 0 |
| Forward | 1 | 1 | 2 | 0 |
| Stop | 0 | 1 | 1 | 2 |
| Sequence 2 | | | | |
| | Left | Right | Forward | Stop |

| | | | | |
|----------------|---|---|---|---|
| Left | 2 | 1 | 1 | 0 |
| Right | 0 | 2 | 2 | 0 |
| Forward | 0 | 1 | 3 | 0 |
| Stop | 1 | 0 | 3 | 0 |

The Binary classifier has an accuracy of 50 % for sequence 1 and 43.75% for sequence 2. The Binary classifier performs worse than the LDA as it does not have a mathematical model on which the probabilities are based

6.3 Wheelchair Response

The wheelchair was not assembled in this project, however the motors were tested at no load speed. This was done to form a general assumption if the motors are capable of frequently changing their speed while having enough output torque to move the weight required.

The software is created in such a manner that the motors reduce their speed over half a second. The ESC is programmed to have a soft start, combined with the software's half second rule. This protects the gearbox from stripping gears in case the acceleration is too fast. It also reduces the amount of bursts of current due to the motor suddenly shifting its speed.

The test setup was monitored with two multimeters and one oscilloscope. The oscilloscope is used to monitor the PWM output sent by the Raspberry Pi to the ESC. The multimeters are used to observe the current each motor draws from the battery.

The entire setup with all its components can be found in the figure below with all the components listed:

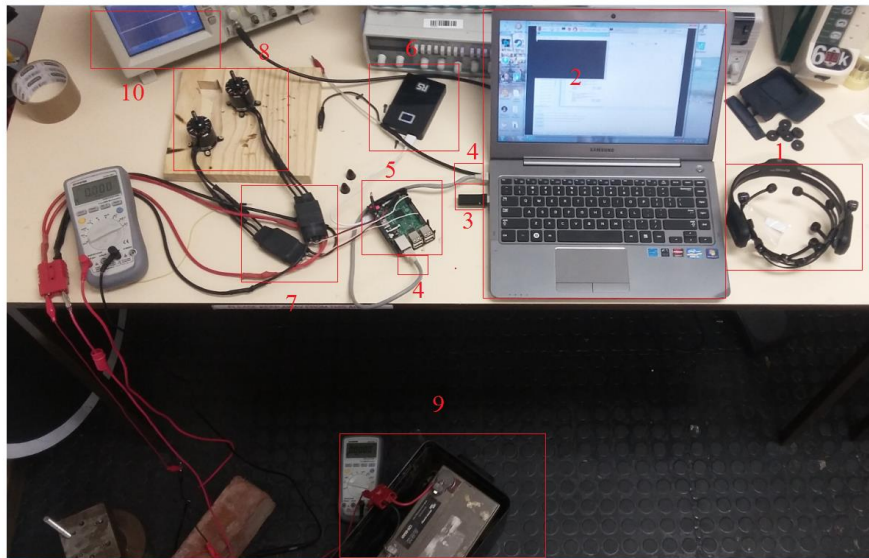


Figure 40: Test Setup of Entire System

The different components in the system are numbered and explained below:

Table 16: Parts and Descriptions of Tested System

| # | Part Name | Description |
|---|----------------|--|
| 1 | Epoc+ | Reads EEG signals from the scalp of user |
| 2 | Laptop | Receives EEG data and computes most likely state. Also sends corresponding value to Raspberry Pi |
| 3 | Epoc+ Dongle | Allows data transition between Epoc+ and laptop |
| 4 | Ethernet Cable | Allows transferring of data between laptop and Raspberry pi |
| 5 | Raspberry Pi | Receives commands from laptop and adjusts PWM outputs accordingly |
| 6 | Power Bank | Used to power up the Raspberry Pi |

| | | |
|----|--------------|--|
| 7 | ESCs | Receives PWM signals from Raspberry Pi and allows motors to turn at right speeds |
| 8 | Motors | Observable components at the end of the chain |
| 9 | Battery | Used to power the motors |
| 10 | Oscilloscope | Allows the monitoring of PWM signals |

The following data was captured on the oscilloscope showing the different pulse widths of each motor for each state.

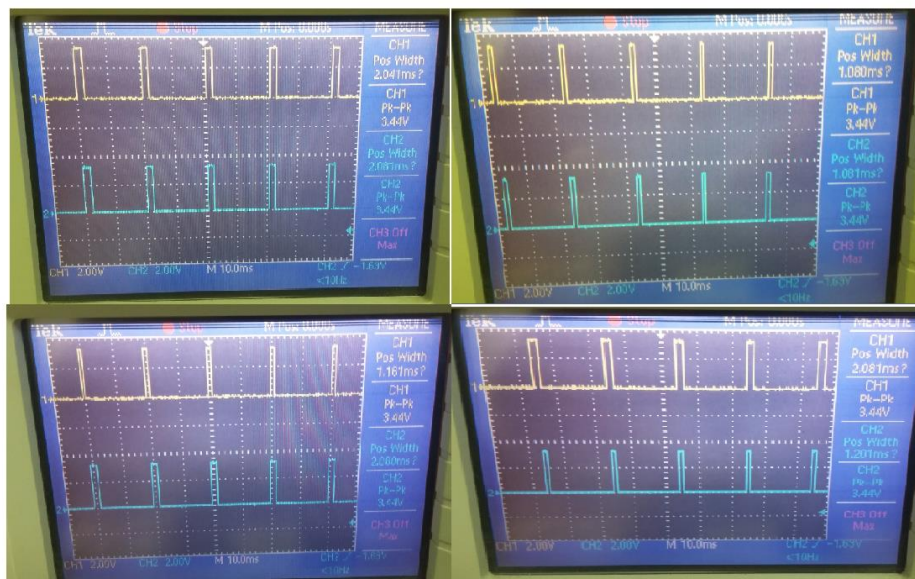


Figure 41: PWM Outputs on Oscilloscope

Where channel 1 and channel 2 are the two different motors. The four states shown above, in order, are: Both motors are moving at full speed, both motors stationary, left motor at full speed and right stationary and left motor stationary and right motor at full speed. The figure is read from left to right and then top to bottom.

Both motors pulled 2.43 A from the battery when at full speed and 0.04 A when stationary.

7. Conclusions and Recommendations

It is concluded that the Binary Classifier is too unstable to be used with the accuracy going below 50%. The main reason for this is that the classifier does not have any mathematical stepping stones. It is purely based on if a value is beyond a certain threshold.

The LDA is also quite inaccurate. Classifications on EEGs can go higher than this result, however they are not all that accurate. Another reason for the inaccuracy is because of the dynamic testing. EEGs that are usually classified are done with pre required data. Dynamic data testing does not have the luxury of time when it needs to be classified and models need to work faster which will deliver less accurate results.

Another observed dilemma is that motor imagery is very reliant on electrode positioning. It is extremely difficult to place the electrodes at exactly the same locations on the scalp as when testing data was recorded. This might cause different patterns for the same thought previously recorded.

The Epoc+ is also not very suited for motor imagery as the electrodes are not placed in steady positions on the scalp, especially with subjects that have thick hair. This can cause some data to be lost because the electrodes aren't close enough to the scalp to record.

The ESC, battery and motor choices are sufficient for the design as the total no load current does not exceed 4.2 A. The main concern is the ESCs which can handle up to 60 A and the motors can draw 80 A each. However the load should not surpass the threshold where the motors are required to draw that much current.

The response time of the system is almost instantaneous, part from the software delays put in to protect the motor and gearboxes. This is an acceptable result as the

data transfer and state changing need to be as fast as possible to reduce unwanted movement or lack thereof.

The price of the entire system without the wheelchair being constructed is far less than an electrical wheelchair, which still need to be modified to accommodate the Epoc+.

The next step would be to manufacture the entire wheelchair and to incorporate a different neuro-headset for testing.

8. References

Aysha, A and Nadia, D. (2016) The Effect Of Anti-Epileptic Drugs On Visual Evoked Potential In Patients With Generalized Tonic-Clonic Seizures: A Prospective Case-Controlled Study. [Online]. Available at https://www.scitechnol.com/peer-review/the-effect-of-antiepileptic-drugs-on-visual-evoked-potential-in-patients-with-generalized-tonicclonic-seizures-a-prospective-casac-VcOT.php?article_id=5525

Brain-Controlled Wheelchair. [Online] 30 July 2016. Available at <http://www.instructables.com/id/Brain-Controlled-Wheelchair/>

Brushless -vs- Brushed DC Motors. [Online]. 14 April 2016. Available at platinumasi.com/Brushless%20Vs%20Brushless%20Motor.pdf

Budynas, R and Nisbett, J. (2015), Shigley's Mechanical Engineering Design. Tenth Edition. New York: McGraw-Hill Education.p 562-597.

Cho, H et al. (2017). EEG datasets for motor imagery brain–computer interface. [Online]. Available at <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5493744/>

Clean Water Team (CWT). (2004). Electrical conductivity/salinity Fact Sheet, FS-3.1.3.0(EC). In: The Clean Water Team Guidance Compendium for Watershed Monitoring and Assessment, Version 2.0. Division of Water Quality, California State Water Resources Control Board (SWRCB), Sacramento, CA

Critical Power. 2009. Victron energy 12v 60ah-55ah agm deep cycle battery. Available at <https://www.criticalpowersupplies.co.uk/12V-60ah-55ah-agm-deep-cycle-battery>.

Demir, G and Oz Mehmet, K. (2005). Online local learning algorithms for linear discriminant analysis. [Online]. Available at <http://www.sciencedirect.com/science/article/pii/S0167865504001825?via%3Dihub>

Emotiv. (2014). Emotiv EPOC & Testbench Specifications. [Online] Available at <https://www.emotiv.com/files/Emotiv-EPOC-Product-Sheet-2014.pdf>

Emotiv (2014) User Manual, Headset and software setup for your Emotiv EPOC neuroheadset [online] Available at https://emotiv.zendesk.com/hc/en-us/article_attachments/200343895/EPOCUserManual2014.pdf

Friction and Friction Coefficients [online]. 30 October 2017. Available at https://www.engineeringtoolbox.com/friction-coefficients-d_778.html

Garcia, et al. (2014). Device and method for cognitive enhancement of a user. EP20120708946

Guitel Black, Red Castor Wheel. Digital Image. Guitel Black, Red Castor Wheels 3588121, 150daN. Web. 6 August 2017. <http://za.rs-online.com>

HK series 50 Amp ESC. Digital Image. Hobbyking™ SS Series 50-60A ESC (Opto only). Web. 12 March 2016. <https://hobbyking.com>

Kirschstein, T and Köhling, R. 1 July 2009. What is the Source of the EEG? [Online] Available at <http://journals.sagepub.com/doi/10.1177/155005940904000305>

LD Power Brushless Outrunner (2820-920Kv). Digital Image. LD Power Brushless Outrunner 35x40mm (2820-920KV). Web. 4 March 2016. <https://hobbyking.com>

Marieb, EN. (2015). Essentials of Human Anatomy & Physiology. Eleventh Edition. Pearson Education. P 252- 263

Membrane Potential vs Time. Digital image. Transmission of Nerve Impulses. 11 November 2017. <https://www.cliffsnotes.com>

Miller, J. (27 August 2010). Brushless Motors vs Brush Motors, what's the difference? [Online] Available at: <https://quantumdevices.wordpress.com/2010/08/27/brushless-motors-vs-brush-motors-whats-the-difference/>

Neustadter, E and Turetsk, B. (2016). EEG and MEG Probes of Schizophrenia Pathophysiology. [Online] Available at: <http://www.sciencedirect.com/topics/neuroscience/p300-neuroscience>

Oostenveld, R Praamstra, P. (2001). The five percent electrode system for high-resolution EEG and ERP measurements. [Online]. Available at: <http://www.sciencedirect.com/science/article/pii/S1388245700005277?via%3Dihub>

P-300 Stimuli. Digital image. 'Mind-Control' gaming devices leak users' secrets. Web. 22 Augustus 2012. Web. 10 November. <http://www.kurzweilai.net>

Raspberry Pi 3B. Digital Image. Raspberry Pi 3 Model B. (17 November 2017). <https://www.raspberrypi.org>

Raspberrypi.org. (2017). Power Supply. Available at <https://www.raspberrypi.org/documentation/hardware/raspberrypi/power/README.md>

Raspberry Pi 3b Pin Layout. Digital Image. Pin Numbering - Raspberry Pi 3 Model B. 26 July 2016. Web. 21 April 2017. <http://pi4j.com>

Relationship between Torque, Speed and Power. Digital Image. Electric Motor Power (Really Simple) and HP Ratings. 9 September 2017. <https://evmc2.wordpress.com>

Relationship between Torque, Speed and Power .Digital Image. Motor Torque Specs Minefield! Web. 18 August 2017. <https://electricbikereview.com>

Somani, S and Shukla, J. (2014). The P300 wave of Event-Related-Potential. [Online]. Available at: <http://www.rroij.com/open-access/the-p300-wave-of-eventrelatedpotential.php?aid=34978>

Sterling, J. (2017). Straight, No Chaser: Brain Health – How Your Brain Works. [Online]. <https://jeffreysterlingmd.com/tag/cerebral-cortex/>

User instruction for RC aircraft ESC. Online. 15 April 2017. Available at <https://hobbyking.com/media/file/1042506394X869309X51.pdf>

Varvel Worm Gearbox. Digital Image. Worm gearbox / right-angle. Web. 21 November 2017. <http://www.directindustry.com>

Visual Evoked Potential (VEP). [Online]. 30 September 2017. Available at <http://sydneyneurology.com.au/visual-evoked-potential-vep>

Visual Evoked Potential. [Online]. 28 September 2017. Available at <https://www.aao.org/bcscsnippetdetail.aspx?id=45cef5ac-2f4e-4b67-81ff-85f3fd02878c>

What is a qEEG? [Online]. 30 October 2017. Available at <http://www.mentetech.com/what-is-a-qeeg/>

What is EEG (Electroencephalography) and how does it work? (2016, February 16th) Available from <https://imotions.com/blog/what-is-eeeg/> [accessed 31 July 2017].

Appendix A: User Manual for the Epoc+

The following steps are used to set up the Epoc+ for data extraction.

1. Wet the electrodes with a saline solution while they are still in the containers.
2. Insert electrodes into sockets on the Epoc+
3. Place Epoc+ on user's head (assuming electrodes are sufficiently wet)
4. Switch headset on at back of device
5. Plug USB dongle into computer
6. Open Pure EEG to check connections

Appendix B: Motor Calculations

The following is an extract from the code used to simulate the amount of time needed to reach maximum speed.

```
interval = 0.1
#change of time each step
time = 0
#current time
torque = 2.622
#current torque
GR = 28
#Gear ratio
alpha = 0.75
#Friction constant
#Gearbox efficiency
mass = 120
#Total mass of system
r = 0.13
#Wheel radius
velocity = 0
#Current Velocity
acceleration = (GR*torque*alpha)/(mass*r)
#Initial acceleration

print "Time = ", time
print "Torque = ",torque
print "Velocity = ",velocity
print "Accel = ",acceleration
print
"*****"
*

#Loop until required speed is reached
while (velocity<3):
    time = time+interval
#Update time
```

```

    velocity =velocity+ acceleration*interval
#Update velocity
    torque = -0.413*velocity + 2.622
#Update torque
    acceleration = (GR * torque *alpha) / (mass * r)
#Update Acceleration

    print "Time = ", time
    print "Torque = ", torque
    print "Velocity = ", velocity
    print "Accel = ", acceleration
    print
    "*****"
    "*"
    "*****"

```

Time = 1.6

Torque = 1.37107135223

Velocity = 3.02888292437

Accel = 1.31833783869

The following graph was used to determine the torque at any given speed or vice versa:

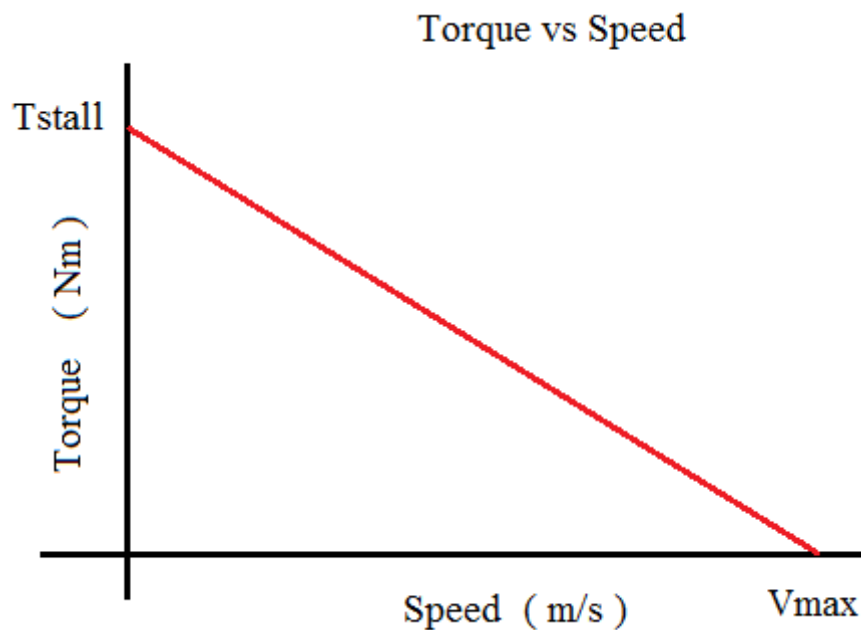


Figure 42: Torque vs Speed

The equation for this function can be show as:

$$T = -\frac{T_{stall}}{v_{max}} \times speed + T_{stall} \quad (B1)$$

Where T_{stall} is the stall torque and v_{max} is the total speed of the motor.

Appendix C Source Code Snippets

The following snippets are from the source code from the classification algorithms and Raspberry Pi.

Appendix C1: Linear Discriminant Analysis

```
def LDA(lda):
    i_data = np.transpose(getNorm('i'))
    l_data = np.transpose(getNorm('l'))
    r_data = np.transpose(getNorm('r'))
    s_data = np.transpose(getNorm('s'))
    f_data = np.transpose(getNorm('f'))

    total = np.concatenate((i_data,l_data,r_data,s_data,f_data))
    labelss =np.zeros(3750)
    for i in range(0,3750):
        if i > 749 and i < 1500:
            labelss[i] = 1
        if i > 1499 and i < 2250:
            labelss[i] = 2
        if i > 2249 and i < 3000:
            labelss[i] = 3
        if i > 2999:
            labelss[i] = 4

    lda.fit(total,labelss)
```

Appendix C2: Binary Threshold

```
def binModel():
    #create labels
    i_data = getNorm('i')
    l_data = getNorm('l')
    r_data = getNorm('r')
    s_data = getNorm('s')
    f_data = getNorm('f')

    l_class=[]
    r_class=[]
    s_class=[]
    f_class=[]

    l_found=0
    r_found=0
```

```

s_found=0
f_found=0

#populate tabel with 1's and 0's
for i in range(0,14):
    l_found = 0
    r_found = 0
    s_found = 0
    f_found = 0
    for j in range(0,768):

        if abs(l_data[i,j]) > 49 and l_found
==0:l_class.append(i);l_found = 1
        if abs(r_data[i,j]) > 49 and
r_found==0:r_class.append(i);r_found= 1
        if abs(s_data[i,j]) > 49 and
s_found==0:s_class.append(i);s_found=1
        if abs(f_data[i,j]) > 49 and
f_found==0:f_class.append(i);f_found=1

    return l_class, r_class,s_class,f_class

```

Appendix C3: Raspberry Pi Code

The following code was used to manually calibrate the ESC and the motors:

```

import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)
GPIO.setup(12,GPIO.OUT)
GPIO.setup(33,GPIO.OUT)

pwm_left = GPIO.PWM(12,50) #pwm (pin, frequency)
pwm_right = GPIO.PWM(33, 50)
pwm_left.start(10)
pwm_right.start(10)          #start(Duty Cycle)

LEFT_SPEED = 10;
RIGHT_SPEED = 10

q= 1

while q:
    time.sleep(0.05)
    str = raw_input()

```

```

if str == '5':
    LEFT_SPEED = 20
    pwm_left.ChangeDutyCycle(20)
    print "Current DC : ", LEFT_SPEED, '%'

    RIGHT_SPEED = 20
    pwm_right.ChangeDutyCycle(20)
    print 'Current DC : ', RIGHT_SPEED, '%'

if str == "2":
    LEFT_SPEED = 10
    pwm_left.ChangeDutyCycle(10)
    print "Current DC : ", LEFT_SPEED, '%'

    RIGHT_SPEED = 10
    pwm_right.ChangeDutyCycle(10)
    print 'Current DC : ', RIGHT_SPEED, '%'

if str == "p":
    print "Connect Betterie"
    x = raw_input()
    pwm_right.ChangeDutyCycle(10)
    time.sleep(6)

mode

if str == "1":
    LEFT_SPEED = 5
    pwm_left.ChangeDutyCycle(5)
    print "Current DC : ", LEFT_SPEED, '%'

    RIGHT_SPEED = 5
    pwm_right.ChangeDutyCycle(5)
    print 'Current DC : ', RIGHT_SPEED, '%'

if str == '0':
    print "Connect betterie"
    x=raw_input()

```



```

pwm_left.ChangeDutyCycle(0)
time.sleep(2)
pwm_left.ChangeDutyCycle(10)
time.sleep(2)
pwm_left.ChangeDutyCycle(5)
time.sleep(2)
print "Armed"

if str == 'j':                                     #Reduce alot
    pwm_left.ChangeDutyCycle(LEFT_SPEED - 0.5 )
    LEFT_SPEED -= 0.5
    print "Current DC : ", LEFT_SPEED," %"

    pwm_right.ChangeDutyCycle(RIGHT_SPEED - 0.5 )
    RIGHT_SPEED -= 0.5
    print "Current DC : ", RIGHT_SPEED," %"

if str == 'g':                                     #Increase
alot
    if LEFT_SPEED <10.1:
        pwm_left.ChangeDutyCycle(LEFT_SPEED + 0.5 )
        LEFT_SPEED += 0.5
        print "Current DC : ", LEFT_SPEED," %"

    if RIGHT_SPEED <10.1:
        pwm_right.ChangeDutyCycle(RIGHT_SPEED + 0.5 )
        RIGHT_SPEED += 0.5
        print "Current DC : ", RIGHT_SPEED," %"

if str == 't' :                                     #Calibrate
ESC
    pwm_left.ChangeDutyCycle(10)
    pwm_right.ChangeDutyCycle(10)
    print "Connect Betterie"
    x = raw_input()
    pwm_left.ChangeDutyCycle(5)
    pwm_right.ChangeDutyCycle(5)
    for i in range (1,13):
        time.sleep(1)
        print (i)
    pwm_left.ChangeDutyCycle(0)
    pwm_right.ChangeDutyCycle(0)

```

```

        time.sleep(2)
        pwm_left.ChangeDutyCycle(5.2)
        pwm_right.ChangeDutyCycle(5.2)
        LEFT_SPEED = 5.2
        RIGHT_SPEED = 5.2
        print "Calibrated !"

    if str == 'q':                                #Exit program
        q=0
        GPIO.cleanup()

    if str == 'd':                                #Reduce      a
little
        pwm_left.ChangeDutyCycle(LEFT_SPEED - 0.05 )
        LEFT_SPEED -= 0.05
        print "Current DC : ", LEFT_SPEED," %"
        pwm_right.ChangeDutyCycle(RIGHT_SPEED - 0.05 )
        RIGHT_SPEED -= 0.05
        print "Current DC : ", RIGHT_SPEED," %"

    if str == 'a':                                #Increase    a
little
        if LEFT_SPEED <10.1:
            pwm_left.ChangeDutyCycle(LEFT_SPEED + 0.05 )
            LEFT_SPEED += 0.05
            print "Current DC : ", LEFT_SPEED," %"
        if RIGHT_SPEED <10.1:
            pwm_right.ChangeDutyCycle(RIGHT_SPEED + 0.05 )
            RIGHT_SPEED += 0.05
            print "Current DC : ", RIGHT_SPEED," %"

```

The next part is used to control the motors given the specific commands:

```

import socket
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)                                #set up pins 12 and 33 for PWM
GPIO.setup(12,GPIO.OUT)

```

```

GPIO.setup(33,GPIO.OUT)

pwm_left = GPIO.PWM(12,50)          #pwm (pin, frequency)
pwm_right = GPIO.PWM(33, 50)         #initialise PWM
pwm_left.start(5)
pwm_right.start(5)                   #start(Duty Cycle)

LEFT_SPEED = 5
RIGHT_SPEED = 5
q = 0

try:
    #Start up server
    print 'Setting up server'
    TCP_IP = '192.168.0.21'
    TCP_PORT = 5005
    BUFFER_SIZE = 20
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind((TCP_IP, TCP_PORT))
    print 'Server Active'
    s.listen(1)
    conn, addr = s.accept()
    print ('connection address', addr)

    #Update duty cycle to fall between max and min specified
    while not q:
        if LEFT_SPEED < 10:
            LEFT_SPEED = 10

        if LEFT_SPEED > 16:
            LEFT_SPEED = 16

        if RIGHT_SPEED <10:
            RIGHT_SPEED = 10

        if RIGHT_SPEED > 16:
            RIGHT_SPEED = 16

        pwm_left.ChangeDutyCycle(LEFT_SPEED)
        pwm_right.ChangeDutyCycle(RIGHT_SPEED)

```

```

        data = conn.recv(BUFFER_SIZE)
        if not data: break
        menu (data)
        print ("received data:", data)
        conn.send(data)

    GPIO.cleanup()
except:
    print 'Server Failed'

conn.close()

def menu(command):
    if command == 'a':
        print 'pressed a'
        right()
    elif command == 'd':
        print 'pressed d'
        left()
    elif command == 's':
        print 'pressed s'
        stop()
    elif command == 'w':
        print 'pressed w'
        forward()
    elif command == ' ':
        print 'pressed <space>'
        q = 1
    else:
        print 'Invalid command'

#stall right motor and increase left motor speed to turn right
def right():
    #reduce and increase speed over 0.5 seconds
    for i in range(0,10):
        if LEFT_SPEED < 8:
            LEFT_SPEED += 0.25
            pwm_left.ChangeDutyCycle(LEFT_SPEED)

        if RIGHT_SPEED > 6:

```

```

        RIGHT_SPEED -= 0.25
        pwm_right.ChangeDutyCycle(RIGHT_SPEED)

    time.wait(0.05)

#stall left motor and increase right motor speed to turn left
def left():
    #reduce and increase speed over 0.5 seconds
    for i in range(0,10):
        if RIGHT_SPEED < 8:
            RIGHT_SPEED += 0.25
            pwm_right.ChangeDutyCycle(RIGHT_SPEED)

        if LEFT_SPEED > 6:
            LEFT_SPEED -= 0.25
            pwm_left.ChangeDutyCycle(LEFT_SPEED)

    time.wait(0.05)

#Stop both motors
def stop():
    #reduce and increase speed over 0.5 seconds
    for i in range(0,10):
        if RIGHT_SPEED > 5:
            RIGHT_SPEED -= 0.3
            pwm_right.ChangeDutyCycle(RIGHT_SPEED)

        if LEFT_SPEED > 5:
            LEFT_SPEED -= 0.3
            pwm_left.ChangeDutyCycle(LEFT_SPEED)

    time.wait(0.05)

#Increase both motors' speed
def forward():
    #reduce and increase speed over 0.5 seconds
    for i in range(0,10):
        if RIGHT_SPEED < 8:
            RIGHT_SPEED += 0.3
            pwm_right.ChangeDutyCycle(RIGHT_SPEED)

        if LEFT_SPEED < 8:

```

```

LEFT_SPEED += 0.3
pwm_left.ChangeDutyCycle(LEFT_SPEED)

time.wait(0.05)

```

Appendix D Engineering Drawings

There are a total of 5 drawings done for this project. The first 4 are key components that are specifically designed to fit with the chosen motors, gearboxes and wheels. The last drawing is the assembly of the different parts. The drawings in order are: Bar, Bracket, Key, Rod and Assembly.

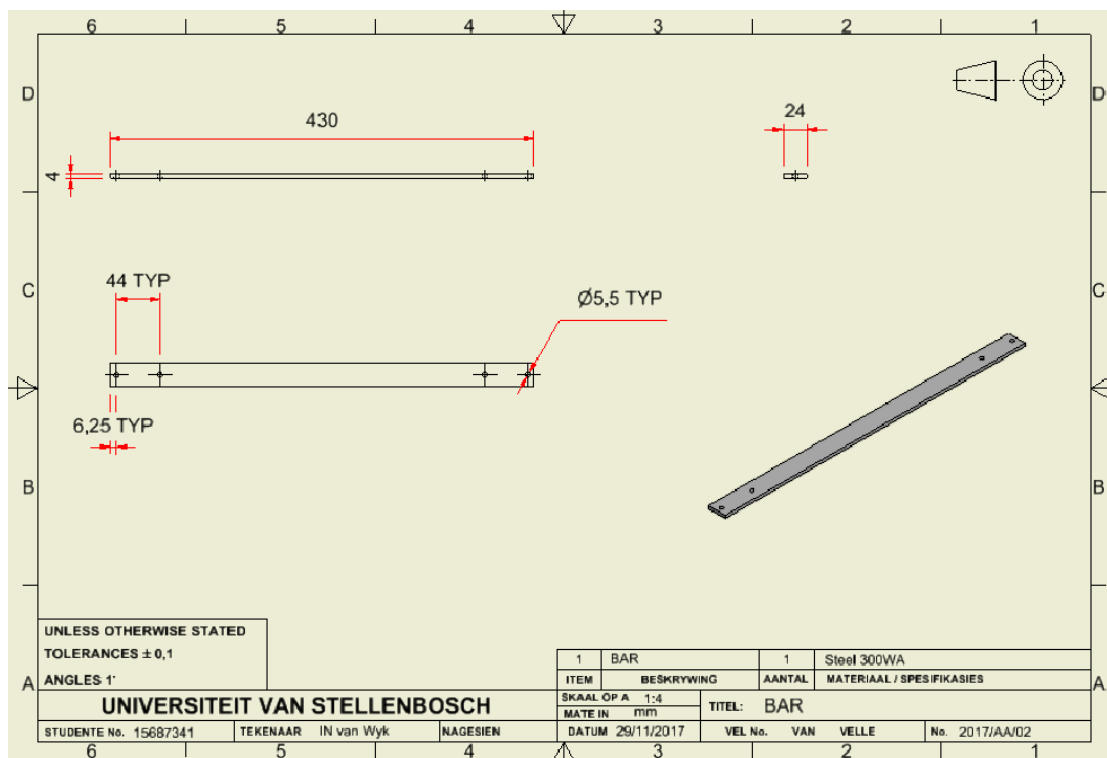


Figure 43: Bar Drawing

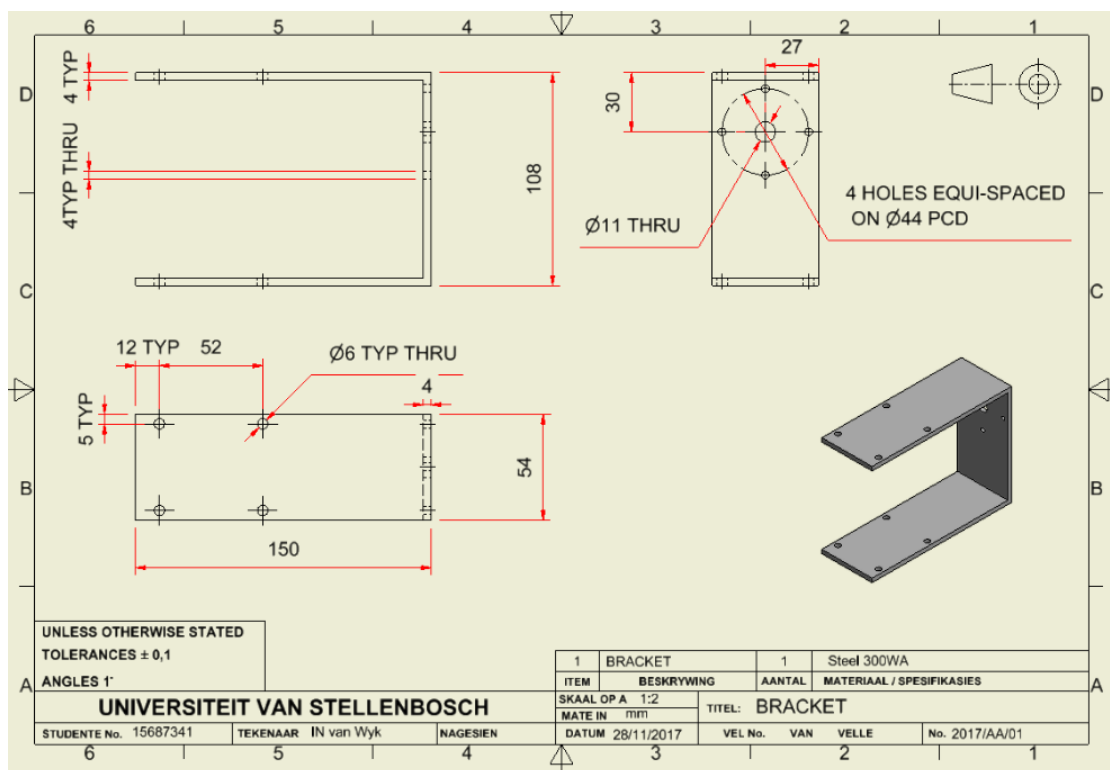


Figure 44: Bracket Drawing

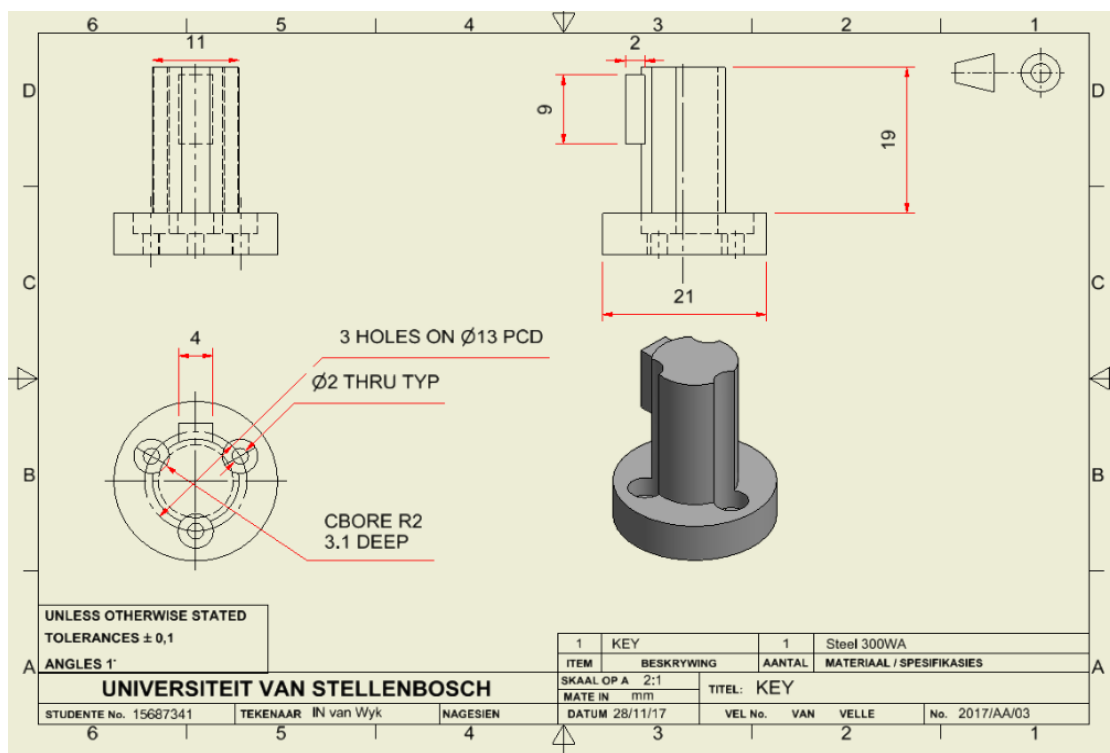


Figure 45: Key Drawing

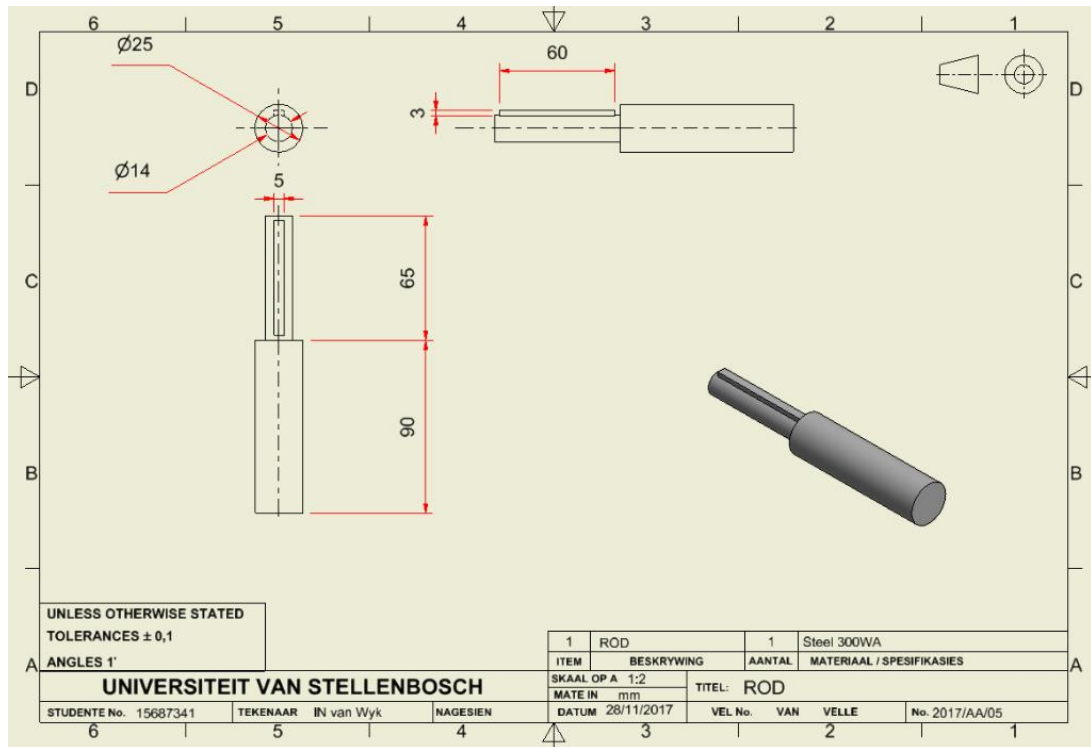


Figure 46: Rod Drawing

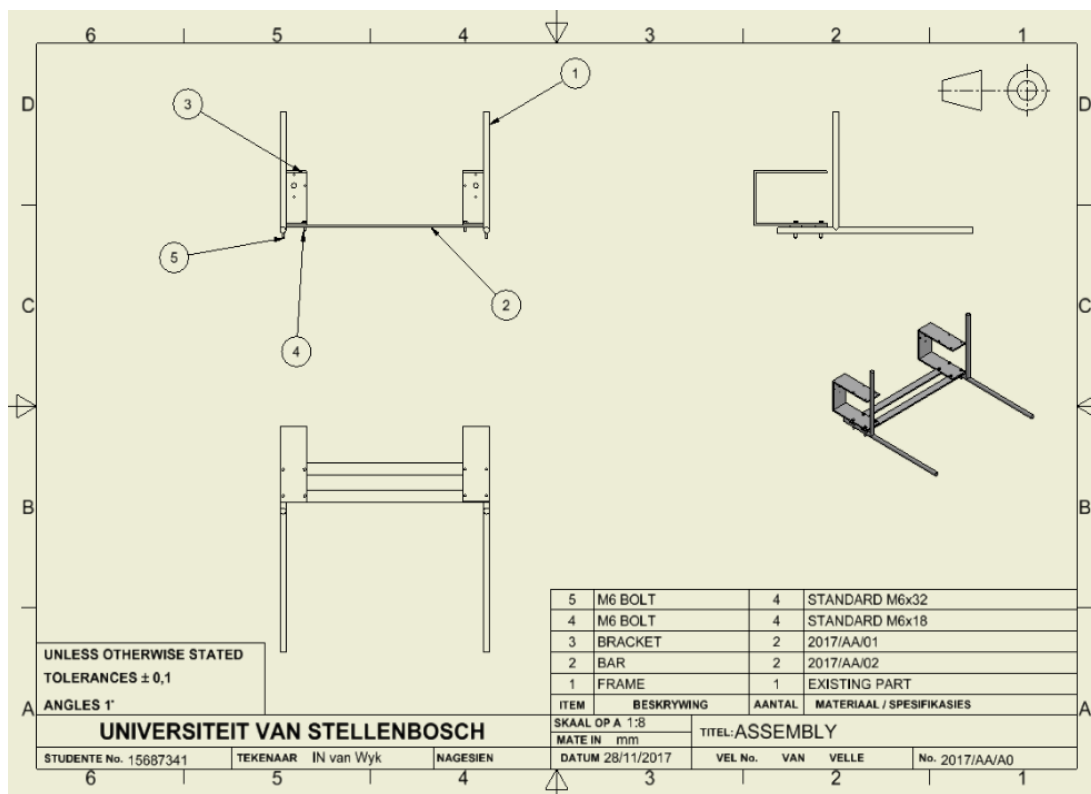


Figure 47: Assembly Drawing

Appendix E List of Programs

Programs used in the project

Raspberry Pi:

- Geany IDE
- Virtual Network Server

Laptop:

- AutoDesk Inventor
- Microsoft Word
- Microsofe Paint
- Microsoft Excel
- PyCharm Developer
- Pure EEG
- Virtual Network Client

Appendix F EEG States

There are a total of 5 states in this project: Idle, Forwad, Left, Right and stop.

Each state is a representation of EEG signals consisting of 14 different channels.

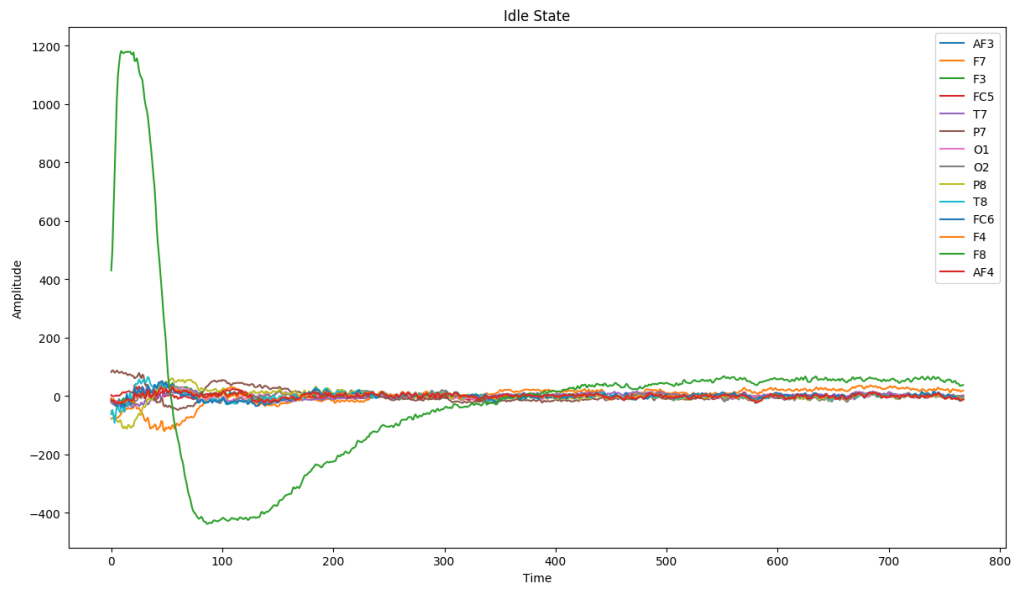


Figure 48: EEG Signals of Idle State

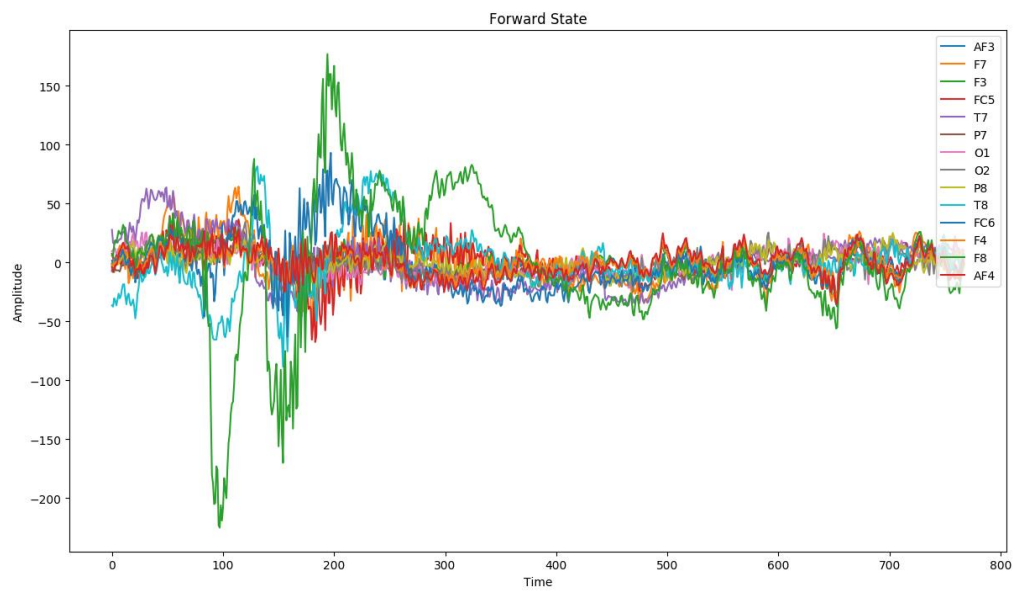


Figure 49: EEG Signals of Forward State

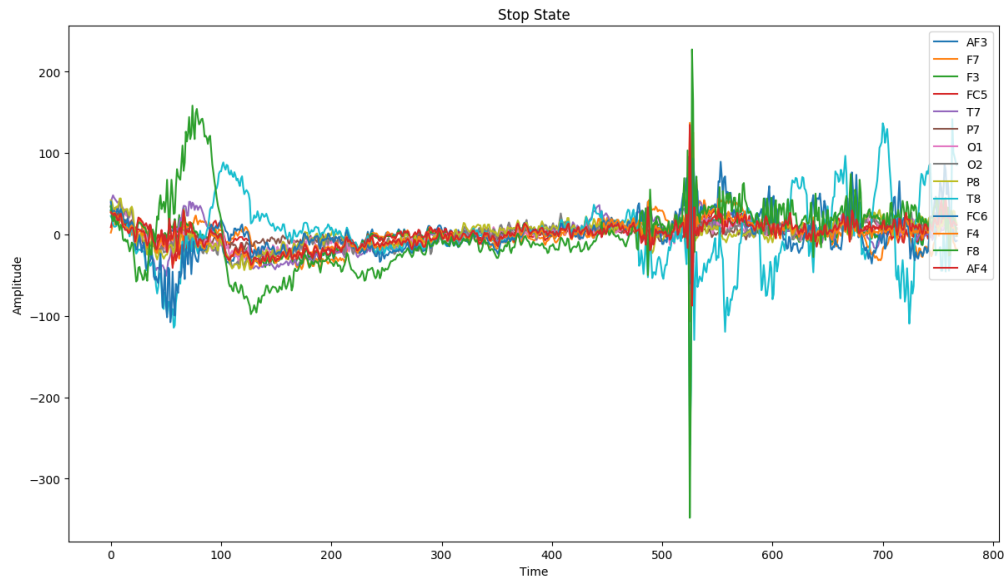


Figure 50: EEG Signals of Stop State

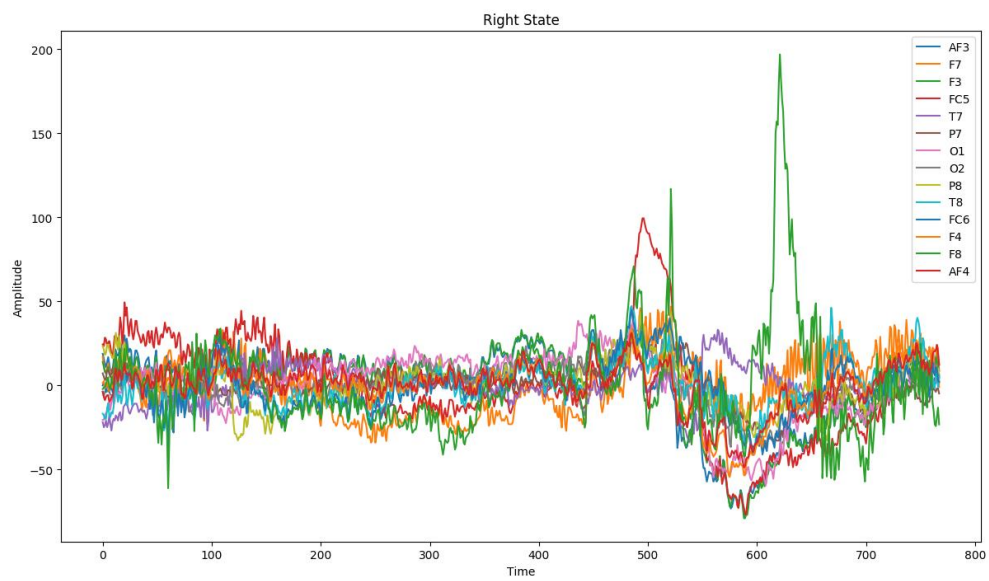


Figure 51: EEG Signals of Right State

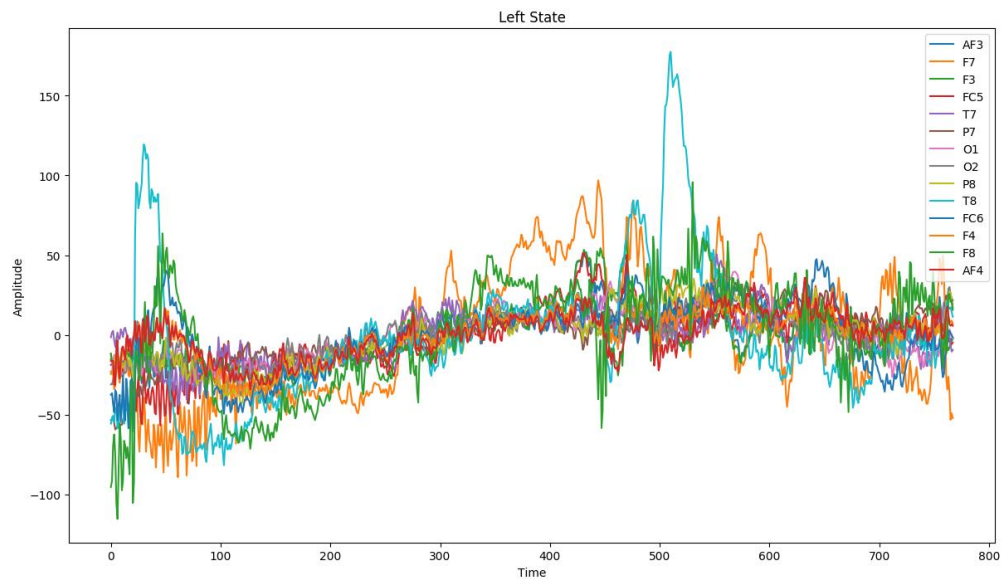


Figure 52: EEG Signals of Left State